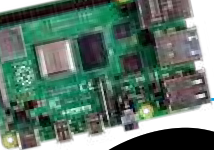


BUY IN PRINT WORLDWIDE MAGPI.CC/STORE

The MagPi

Issue 111 | November 2021 | magpi.cc

The official Raspberry Pi magazine

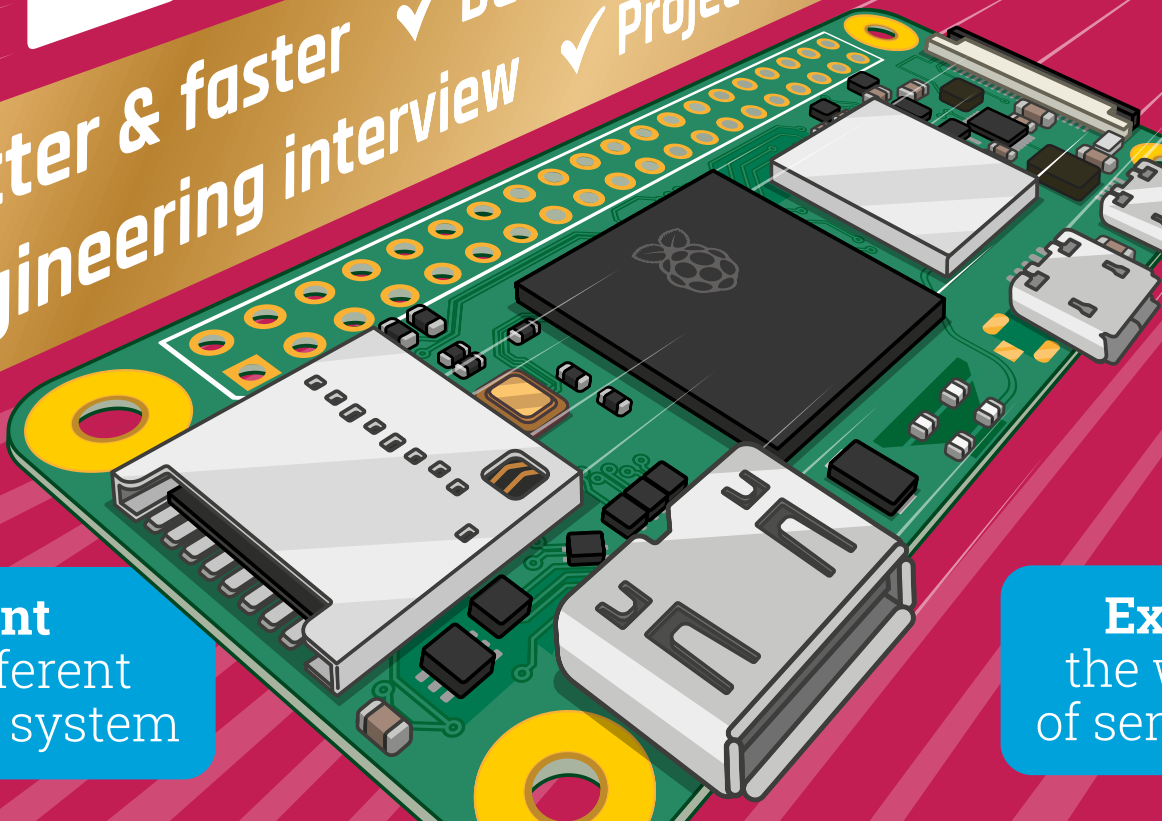
NEW! RASPBERRY PI ZERO 2

GET YOURS FREE!

SEE PAGE 32

✓ Better & faster ✓ Detailed specs

✓ Engineering interview ✓ Project ideas



Experiment with a different operating system

Explore the world of sensors!

BUILD A RASPBERRY PI PICO STREAM DECK

WELCOME
to *The MagPi* 111

The story of Raspberry Pi Zero is intertwined with *The MagPi* magazine. Zero was first introduced to the world in *The MagPi* magazine issue #40. This fantastic single-board computer is the entry point for many of our readers.

We're impossibly proud to introduce Raspberry Pi Zero 2 W. Thanks to a clever design technique, it packs a faster quad-core CPU into the same tiny footprint. You can use Zero 2 in your current project (just swap out a Zero for a Zero 2 and get an instant speed boost), and Zero 2 is compatible with the vast array of kits and current home builds.

Our Zero 2 feature (page 34) brings you all the information on this exciting new computer. And every subscriber to *The MagPi* in print will be getting a free Zero 2 in the post. We take great care of our subscribers and routinely hand out free gifts, from the original Raspberry Pi Zero to Google AIY Voice Kits, Raspberry Pi OS installation CDs and cooling stands, and now, Zero 2.

It is worth subscribing to *The MagPi*. All new subscribers will also get a Raspberry Pi Zero 2 W (see page 32). On a personal note, a big thank-you to everybody who subscribes to *The MagPi*. You keep this magazine alive.

Lucy Hattersley Editor

EDITOR

Lucy Hattersley

Lucy is editor of *The MagPi* and it's months like this that she wishes we'd done the issue numbering in binary.

magpi.cc

GET A RASPBERRY PI ZERO 2 W WITH A SUBSCRIPTION! PAGE 32

Contents

> Issue 111 > November 2021

Cover Feature

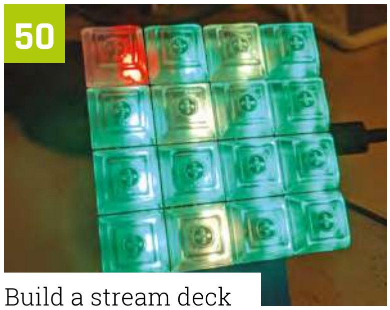
Regulars

Project Showcases

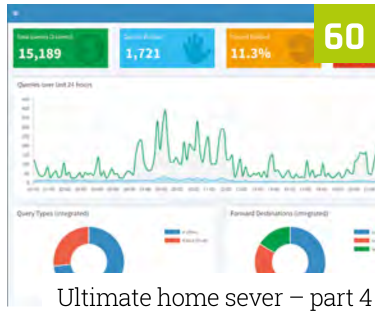
Automatic Guitar Tuner

Community Jams

Tutorials

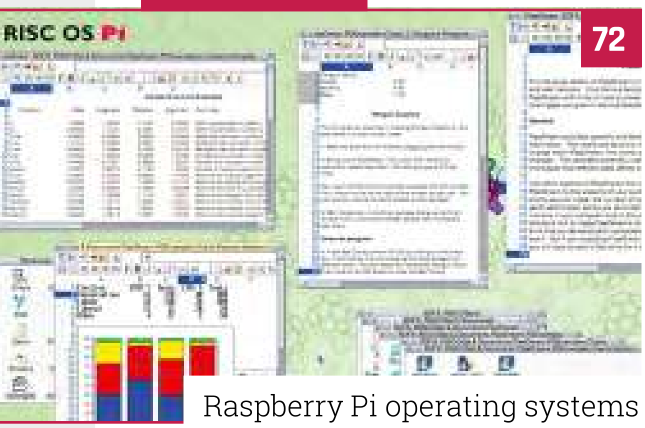


Build a stream deck

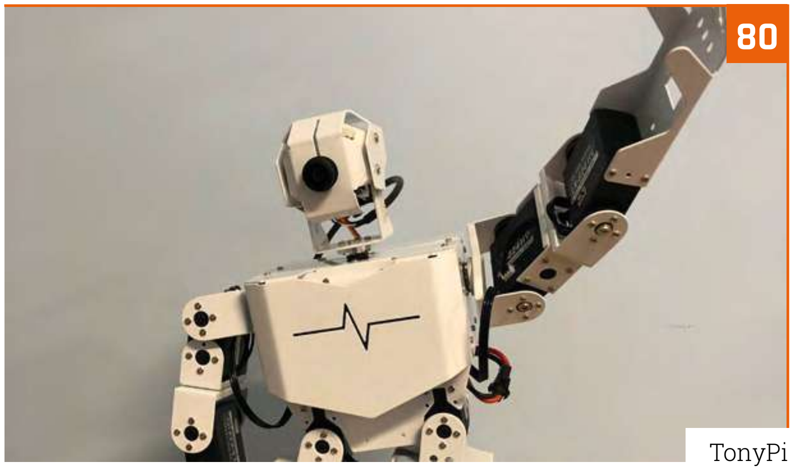


Ultimate home sever – part 4

The Big Feature

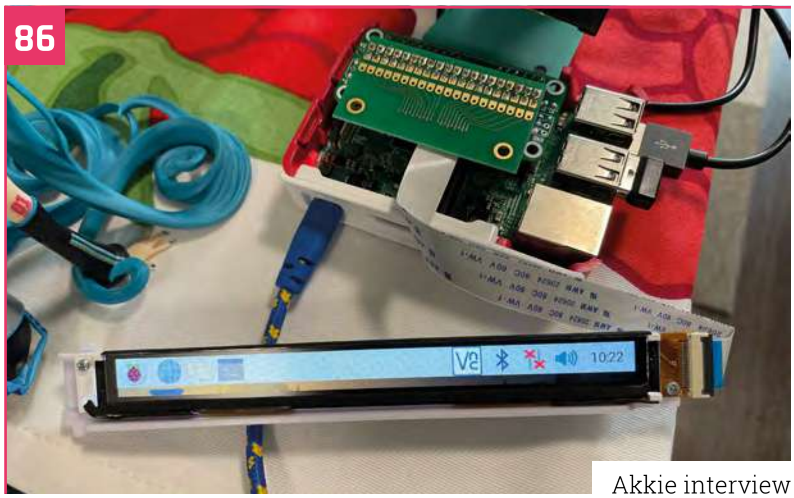


Raspberry Pi operating systems



TonyPi

Reviews



Akkie interview

Community

WIN 1 OF 5 WAVESHARE E-INK **DISPLAY PHAT** 95

Raspberry Pi Build HAT announced

Discover the new collaboration between LEGO® Education and Raspberry Pi. By **Lucy Hattersley**

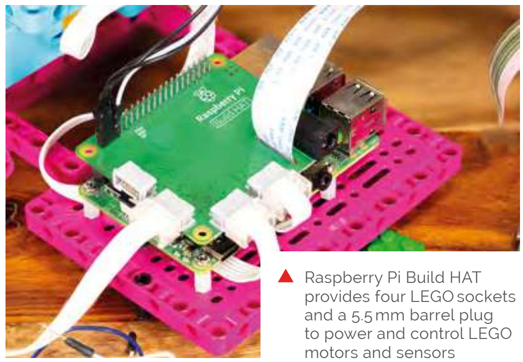
Raspberry Pi and LEGO® have been collaborating on a new product: the Raspberry Pi Build HAT.

The Build HAT is designed to combine the power of Raspberry Pi with the LEGO Maker Plate included with the LEGO Education SPIKE™ Prime expansion set.

Tom Hall, General Manager at LEGO Education, said: "Combining SPIKE Prime with Raspberry Pi will inspire students to explore more advanced creations and continue to nurture their interest in STEAM learning. This brings in even more high ceiling learning opportunities that tap into the maker space and encourage students to combine different products. The Raspberry Pi Build HAT will be available through Raspberry Pi's authorised reseller network."

Brick building

The Build HAT is compatible with the most recent generation of LEGO Technic™ motors, and sensors. An optional extra, the Maker Plate, is included in the latest LEGO Education SPIKE Prime portfolio (version 45681; £362 / $340; **magpi.cc/spikeprime**). With the Build HAT attached, Raspberry Pi can control up to four LEGO motors and/or sensors using Python code.



▲ Raspberry Pi Build HAT provides four LEGO sockets and a 5.5 mm barrel plug to power and control LEGO motors and sensors

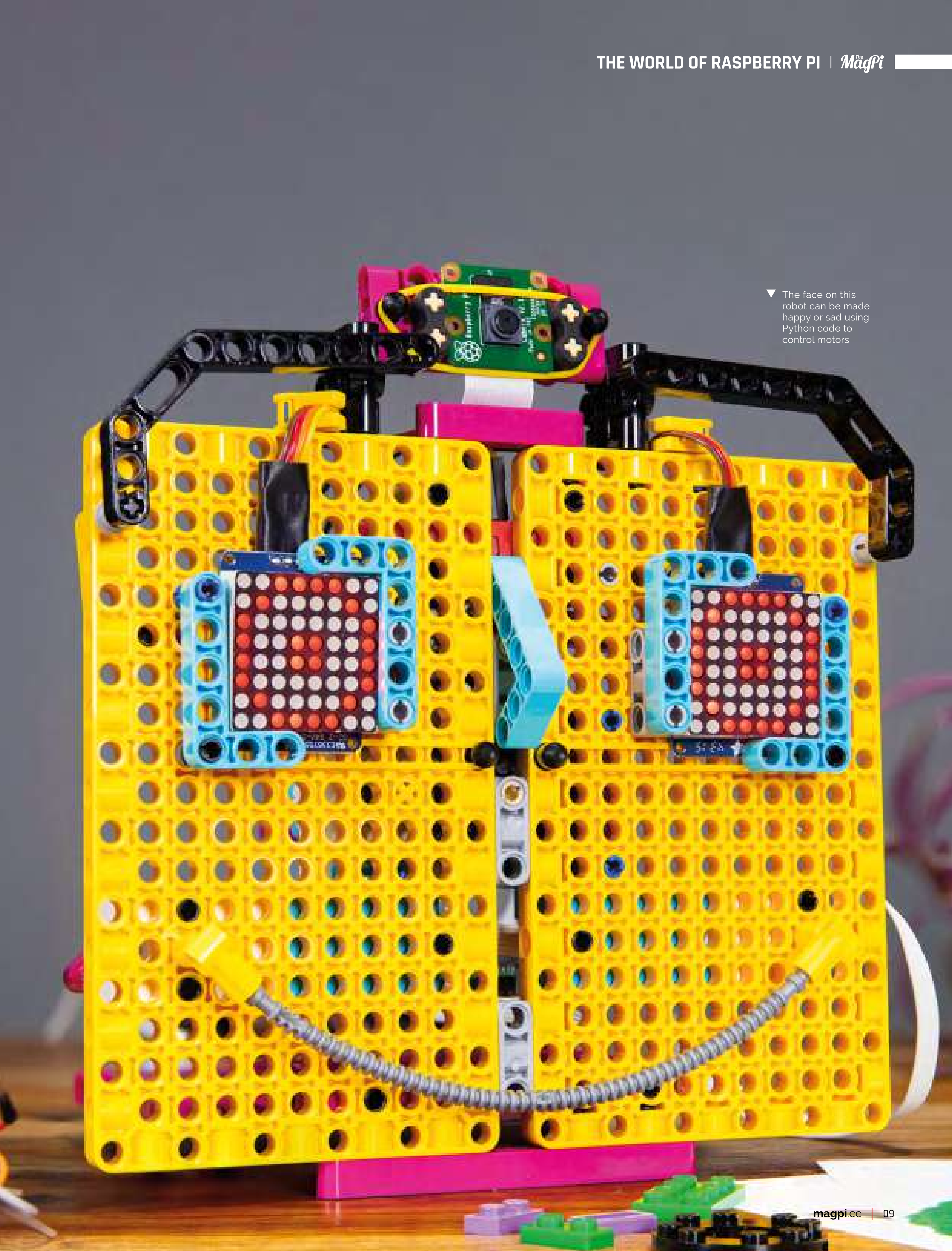
> "Combining SPIKE Prime with Raspberry Pi will inspire students"

The Build HAT itself costs $25 and a compatible 48 W (8 V DC, 6 A) power supply is sold alongside for $15. With the power supply attached, the Build HAT powers both Raspberry Pi and LEGO motors and sensors.

Build HAT is the second Raspberry Pi product to feature the RP2040 microcontroller designed by Raspberry Pi and first introduced with Raspberry Pi Pico.

See **magpi.cc/buildhat** for more information.

▼ Build a remote control car with Raspberry Pi and LEGO



▼ The face on this robot can be made happy or sad using Python code to control motors

Photo credit: ELBA

The kit comprises a Raspberry Pi 400 Desktop Kit and monitor

Leaving a legacy

IT veteran **Gerry Fillery** leaves a donation to help the next generation of coders

While Raspberry Pi was initially developed for young people, folks of all ages have picked up our favourite microcomputer for fun and practical projects. Gerry Fillery was one of those people.

Having retired from his career in IT ten years ago, Gerry took an interest in Raspberry Pi, and had projects set up around his house. "Each time I would visit him he seemed to have acquired more bits and pieces, he was quite an enthusiast," recalls his friend.

Gerry's interest in Raspberry Pi was so great that he decided to leave a donation to the Raspberry Pi Foundation in his will, and is the first person to have done so. His generosity will help support the computer science and maker education of millions of young people all over the world.

Connecting people

A part of this donation has gone towards distributing free Raspberry Pi Desktop kits to those who do not have a computer at home as part of the Stay Connected to School initiative, allowing them to stay connected to learning whilst schooling is disrupted.

> "His generosity will help support the computer science and maker education of millions of young people"

"Gerry is our first ever legacy donor," said Teresa Hicks, Director of Strategic Partnerships at the Raspberry Pi Foundation. "We are incredibly grateful for his generous gift. His legacy lives on through helping us to bring maker education to young people all over the world".

Vocal vocation

Gerry grew up in Blackheath, London. As a youngster, he also had links to Fairlop and Seven

The kit comprises a Raspberry Pi 400 Desktop Kit and monitor

Donate to the Raspberry Pi Foundation

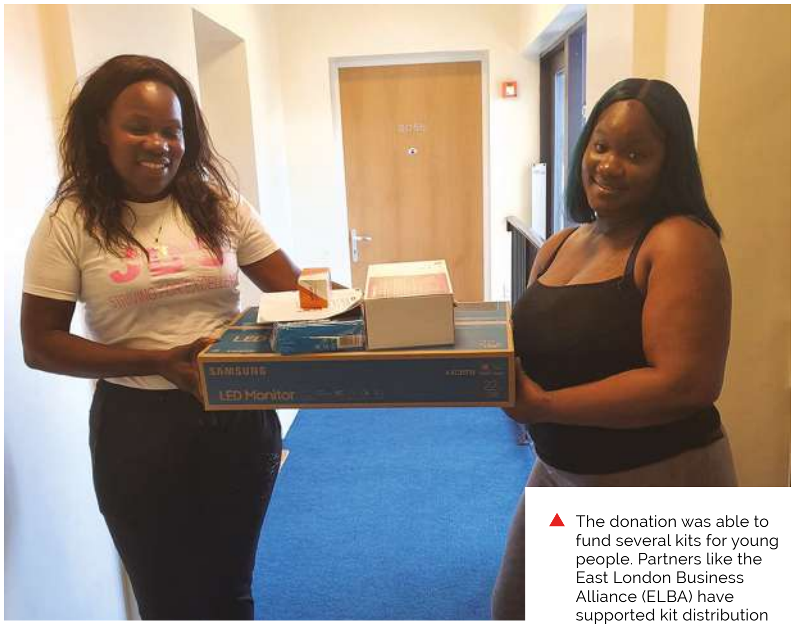
If you want to follow in Gerry's footsteps and help bring computers and computing education to young people around the world, you can donate to the Raspberry Pi Foundation at **rpf.io/donate**.

Kings, and was educated at Edinburgh University where he studied as a Statistician and Actuary.

He worked as a techie throughout his career, preferring a technological challenge.

He would often talk of voice applications in tech, a field he was very familiar with: he worked on early bank telephone automation systems.

He lived for the last 20 years of his days in Frimley Green, Surrey. Thank you, Gerry, for all you have done.



The donation was able to fund several kits for young people. Partners like the East London Business Alliance (ELBA) have supported kit distribution

SPY-DER Robot

A multi-legged surveillance robot combines AI and Raspberry Pi. **Rosie Hattersley** learns some spidey-sense

MAKER

Arijit Das

Indian computer science engineer Arijit runs a group of like-minded tech enthusiasts called Sparklers that focuses on making AI- and robotic-based projects such as a Sudoku-solving buddy.

magpi.cc/spyder

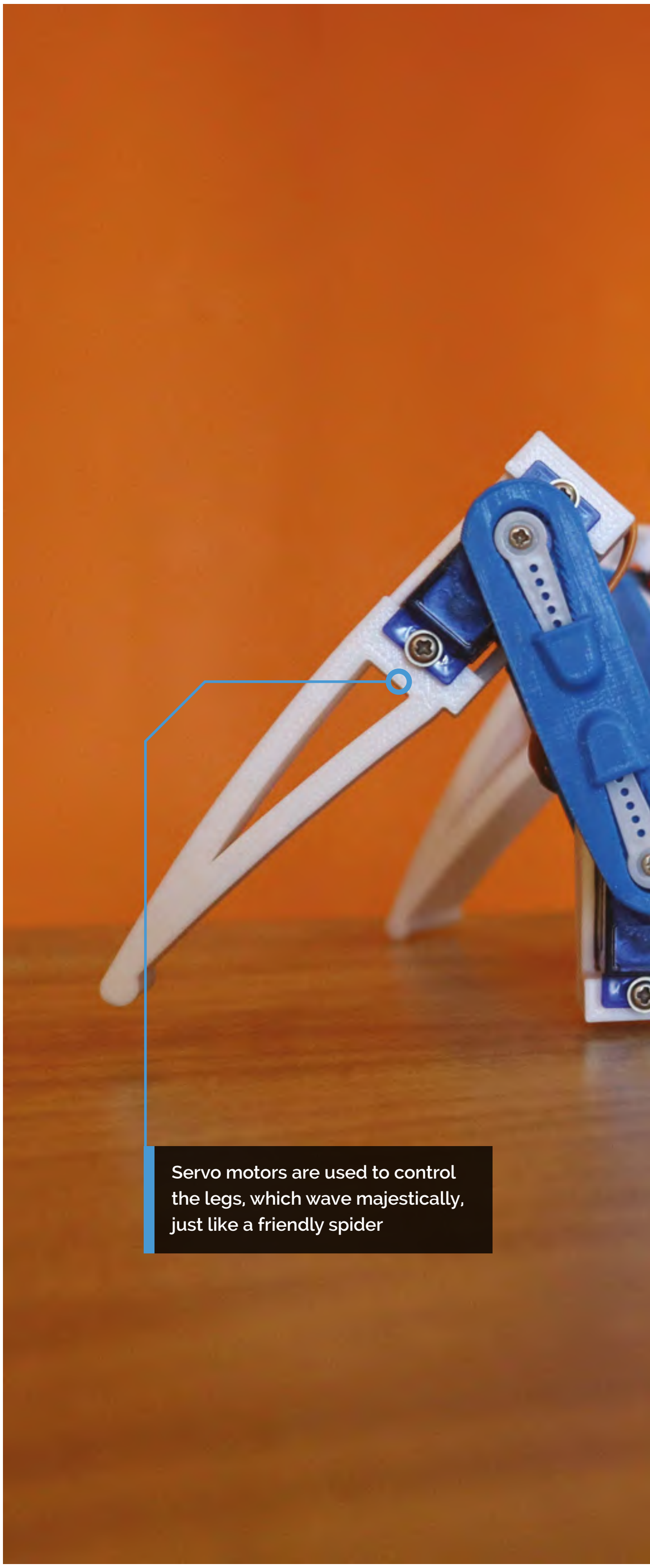
Those of us with an aversion to the company of spiders may find themselves a little uncomfortable at the design of the latest AI robot created by the Sparklers, a group of computer science engineers in India. Not only does the robot specialise in watching its object's every move, but it has multiple legs and the ability to move in unexpected ways.

Team leader Arijit Das explains that SPY-DER is so-named because of its ability to walk like an arachnopod – much to the consternation of some observers. The robot's main purpose is for live video surveillance, streaming footage that can be viewed remotely via a dedicated web browser. SPY-DER can be controlled in two ways, either through voice commands or from the web interface. Riffing on one of the favourite items of prey for real-life spiders, Arijit uses 'bumblebee' as the robot's wake word, prompting its LED eyes to light up.

"Whenever I call it by its name, it starts listening to me and then, based on my voice command, it will act," he clarifies. These commands cover both speech recognition and 'intent detection'. "I can give SPY-DER the same command in different ways, i.e. 'Wave your hands' or 'Say hello'. Both commands make it wave its legs," says Arijit. The online control panel, meanwhile, lets the user control all of SPY-DER's actions. If such movements don't make you tense up, you can take a look at SPY-DER in action at **magpi.cc/spyderyt**.

Evolutionary process

The first version of SPY-DER was an Arduino Nano-based device, controllable via an Android smartphone



Servo motors are used to control the legs, which wave majestically, just like a friendly spider

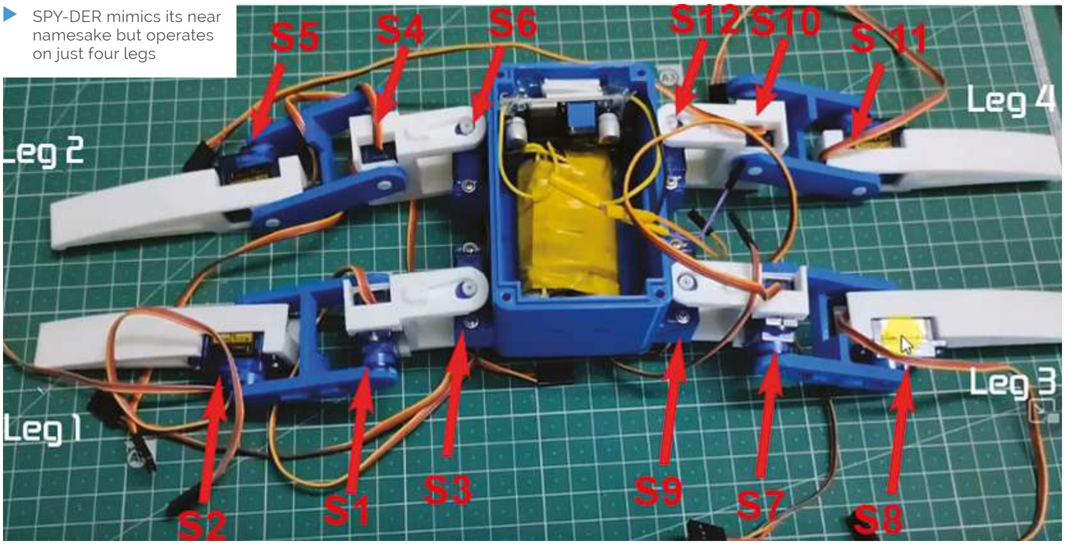
Raspberry Pi Zero W enables the web control interface and manages the voice recognition and AI elements

Don't be alarmed, but on hearing the wake word, the SPY-DER robot's LED eyes light up

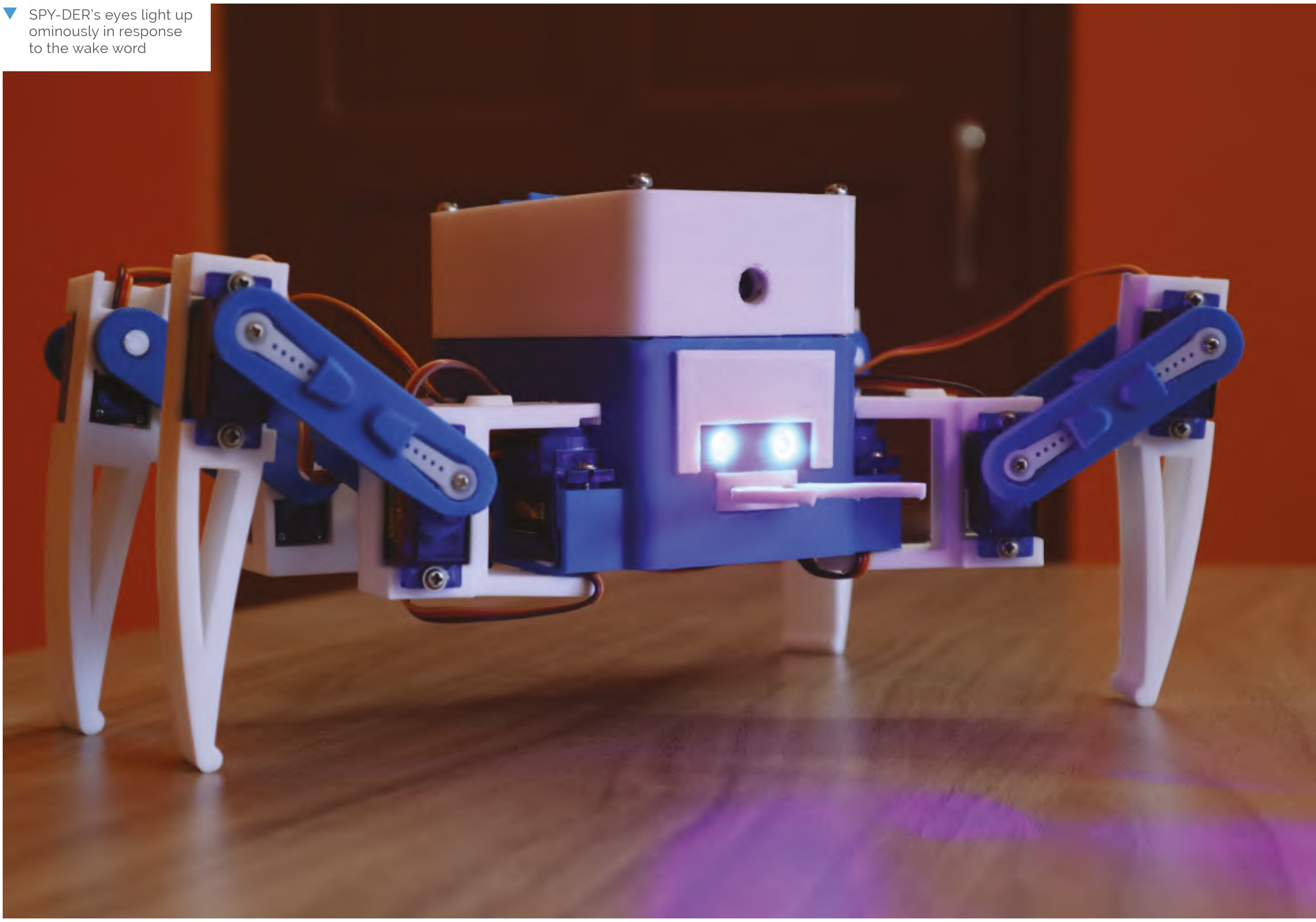
Quick FACTS

- SPY-DER can be built over the course of a week
- Arijit's version of the chassis is here: **magpi.cc/spyderprint**
- He built his own after seeing how much commercial robots cost
- Reactions from friends and his pet rabbit have been extreme…
- Extreme delight or extreme fright!

SPY-DER mimics its near namesake but operates on just four legs



SPY-DER's eyes light up ominously in response to the wake word



Arijit 3D-printed the parts for the spider robot's chassis

or web browser, but lacking the voice recognition aspect Arijit was keen to add. "Obviously I needed a small computer here," he comments. Several of his previous projects, including the Sudoku-solving robot we featured in *The MagPi* #98 (**magpi.cc/98**), were based around Raspberry Pi. Size constraints led Arijit to choose Raspberry Pi Zero W this time. Wireless connectivity and the ability to connect to a Raspberry Pi Camera Module were also critical features.

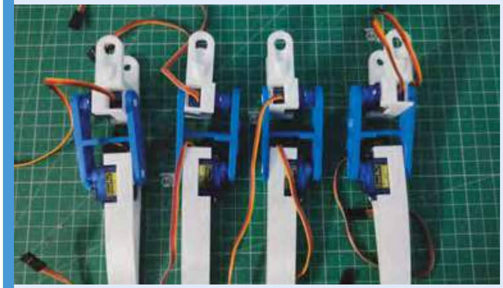
The limited RAM provision of Raspberry Zero W presents a challenge for Arijit's hope of eventually implementing a local speech and intent recognition system from scratch. The speech recognition system would need to be highly optimised, otherwise it will take a lot of time to recognise the speech, he acknowledges. For now, SPY-DER uses an amended version of Picovoice (**picovoice.ai**) speech recognition, but in time Arijit also aims to add image processing and AI-based features, such as object tracking and face recognition.

Building it

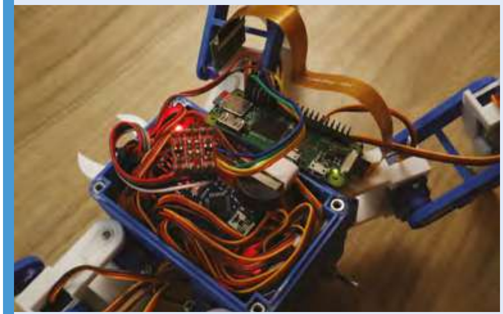
Rather than designing SPY-DER's body from scratch, and having recently got his first 3D printer, Arijit adapted an existing spider robot chassis (**magpi.cc/spiderbot**), changing its dimensions to accommodate Raspberry Pi Zero W, microphone, and the Camera Module. He tweaked the spider robot's Arduino code, but wrote the Python code for Raspberry Pi to be able to control SPY-DER. The web controls use a Flask framework with a web page coded in HTML, CSS, and jQuery. "For the live video streaming, I used RPi-Cam-Web-Interface (**magpi.cc/rpicamweb**), due to the fact that the latency is very low here," Arijit explains. Helpfully, he provides diagrams and code at **magpi.cc/spyder**.

I spy… spy spiders

To build your own spider robot, first follow Arijit's online instruction video: **magpi.cc/spyderbuild**. The second YouTube video explains how to incorporate Raspberry Pi in order to make use of AI features: **magpi.cc/spyderai**.



01 Download and 3D-print the spider body parts (**magpi.cc/spyderprint**). Then attach the servo motors for the legs, and assemble the rest of the body.



02 Attach battery and charging cables, Arduino and Arduino IO shield, and install the Arduino code before inserting Raspberry Pi Zero W, USB microphone, and Camera Module.

03 Download the project code from Arijit's GitHub repository (**magpi.cc/spyder**) along with Picovoice, amending the code as indicated. For video streaming, you'll also need to install the RPi-Cam-Web-Interface.

Automatic Guitar Tuner

A Pico-powered automatic tool to tune your guitar? **Nicola King** is inspired to start strumming

MAKER

Guyrandy Jean-Gilles

Guyrandy is an engineer and maker from the US east coast. He got his bioengineering degree from the University of Pennsylvania, and currently works in the medical device industry.

magpi.cc/tunergitlab

Let's face it, there's no point in having an ultra-cool guitar if you don't have the means to make sure it stays in tune, especially if you're pursuing a Springsteen-esque form of brilliance on the fretboard.

Guyrandy Jean-Gilles is a musical maker who took Raspberry Pi Pico and created something rather special. "The goal of the project was to make a low-cost, automatic guitar tuner," he tells us. "I'd been meaning to do a project with Pico and there are few development boards as cheap and well-documented."

Pico power

Guyrandy developed his idea over a two-month period, and the result is a very useful device. The guitarist chooses whichever string they want to tune and then places the tuner on the appropriate tuning peg. A button on the automatic tuner is held down by the user which then activates the microphone, and the tuner begins to determine the pitch of the tone coming from the plucked string. It then twists/adjusts the peg as necessary to ensure accurate tuning for sweet sounds.

Guyrandy tells us that Pico is the brains of the entire tuner: "It samples an electret microphone at 4096 samples per second, computes an FFT (fast Fourier transform), finds the strongest magnitude frequency in the audio, then turns a DC motor clockwise or counter-clockwise until the target frequency and the strongest frequency in the audio match."

> I'd been meaning to do a project with Pico and there are few development boards as cheap and well-documented

Change your tune

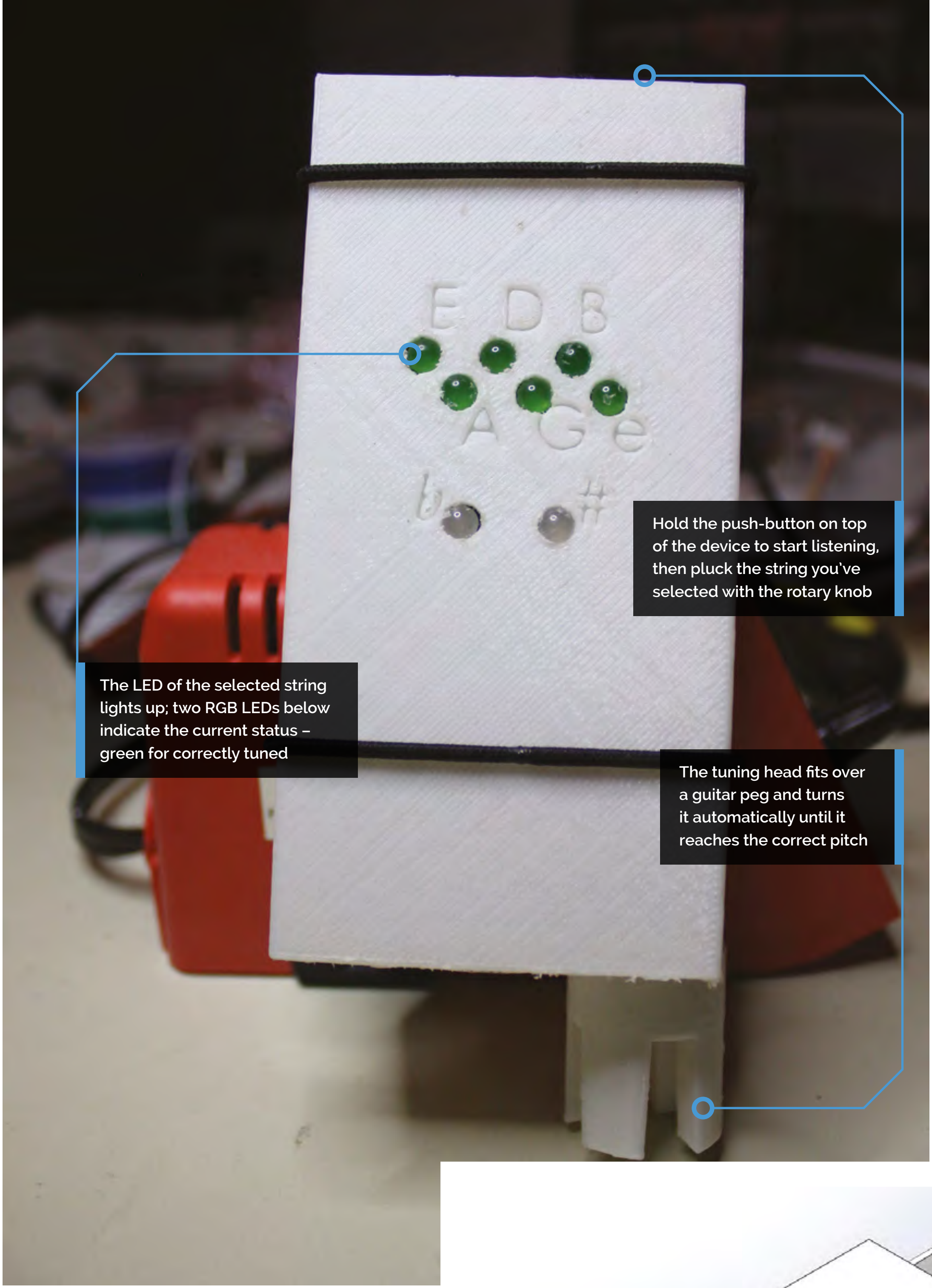
As with all electronic makes, this was a learning experience with a few bridges to be crossed, and the trickiest part of the build was determining a guitar string's pitch from audio. "There's a lot of academic research in the area that I wasn't aware of before starting the project," says Guyrandy. "This version found the strongest magnitude in a fast Fourier transform, and assumed that was the fundamental frequency."

In terms of accuracy, he found a few blips that he had to even out. "The tuner's frequency detection is repeatable but inaccurate. It will mistakenly think harmonics are the fundamental frequency for almost all cases. I had to hard-code harmonic frequencies into the firmware to make the tuner work appropriately," he explains.

That said, the tunings are reliably inaccurate so they can be used to correctly tune a guitar. "For example, the low E string is 82.4 Hz in standard tuning, but the project repeatable thinks it's 250 Hz. Currently alternate tunings aren't possible, but with a firmware change, drop tunings can be achieved."

Guyrandy has had plenty of suggestions from other makers in terms of potential improvements, including adding a vibration sensor so the tuner can be used in a noisy setting, and also "to make a tuner that fits over all the tuning heads of the guitar and tunes all the strings simultaneously.

Guyrandy demonstrates his automatic guitar tuner, here using it to turn the peg for the top E string



Hold the push-button on top of the device to start listening, then pluck the string you've selected with the rotary knob

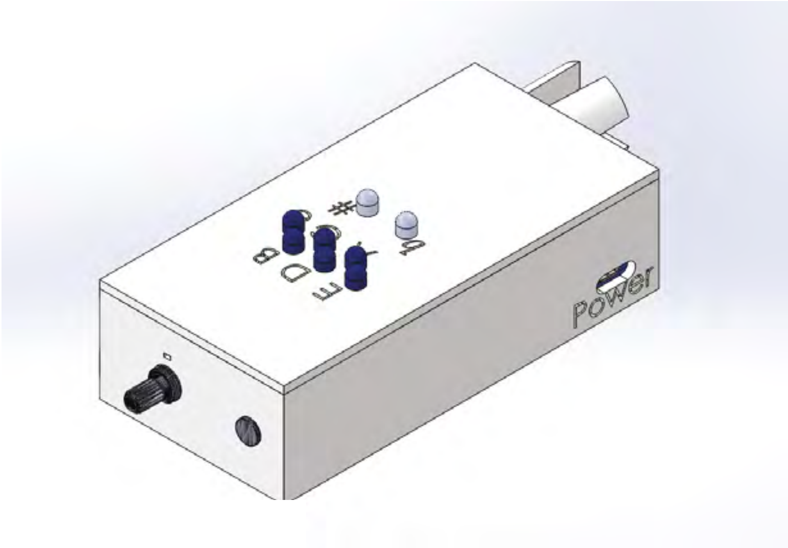
The LED of the selected string lights up; two RGB LEDs below indicate the current status – green for correctly tuned

The tuning head fits over a guitar peg and turns it automatically until it reaches the correct pitch

Quick FACTS

- Guyrandy was inspired by the Roadie 3 guitar tuner: **magpi.cc/roadie3**
- To make your own, you'll need a Pico, Adafruit PowerBoost 1000…
- …plus tactile buttons, mini metal gear motor, and some LEDs
- In 2017, Guyrandy snappily decorated his graduation cap and gown…
- …with the official 7-inch display and a Raspberry Pi 3: **magpi.cc/gradcap**

Guyrandy designed a slick-looking case for the tuner in CAD software and 3D-printed the parts for it



While this is cool, I'm not sure how I'd separate two different fundamental frequencies from one audio signal."

He is currently working on a second version of the tuner and will be trying the YIN (**magpi.cc/yin**) algorithm to detect frequency, as well as incorporating some piezo-electric sensors to pick up vibrations, along with a stronger motor.

If you'd like to have a go at making your own version of the tuner, Guyrandy has generously made his code open-source, and information on exactly what you need in order to build it can be found on his GitLab page (**magpi.cc/tunergitlab**). Why not dust off that old guitar and get making?

Electronic nose

Sniffing out your surroundings with this next step in sensor technology. **Rob Zwetsloot** asks a Raspberry Pi to detect cheese or petrol

MAKER

Luis Rodriguez Mendoza

A Geomatics Engineering Master's student at the University of Calgary, he is passionate about health and science, and applications of wearable technology.

Light sensors, temperature sensors, humidity sensors, noises sensors (i.e. microphones) are all fairly common, however gas sensors are not something you see everyday. Using them specifically to 'smell' your surroundings is even rarer. Luis Rodriguez Mendoza is the creator of one of these rare projects.

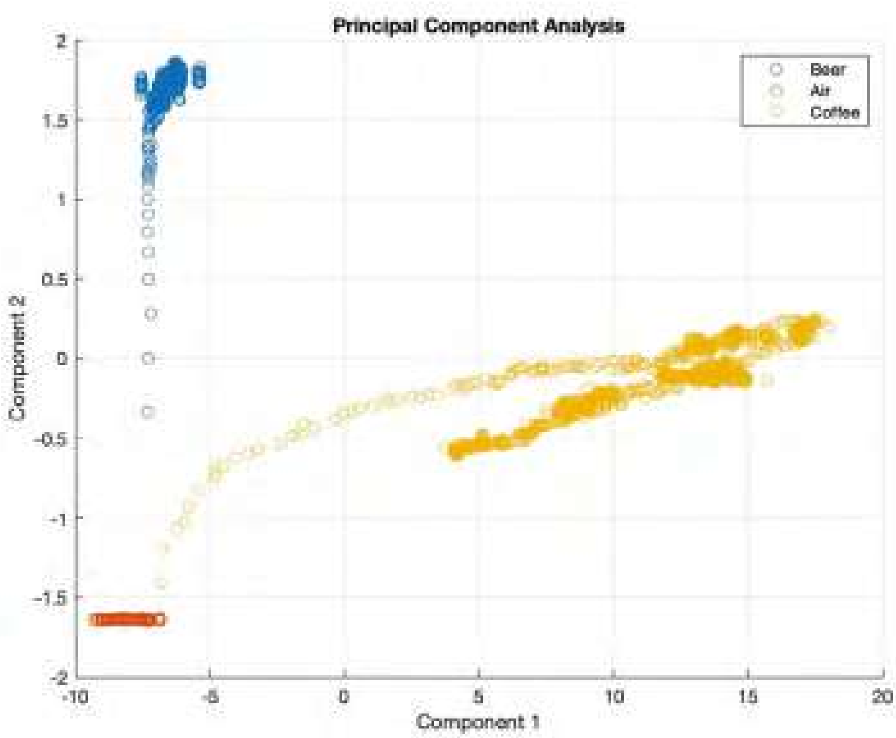
After learning about the different and weird things dogs have been trained to identify by smell at the airport, Luis asked the question of whether he could do something similar with low-cost gas sensors.

"The purpose of the project is to show that low-cost sensors can be reliable in detecting odours and that they can possibly be used in clinical settings." Luis tells us. "Testing was done using samples of beer and brewed coffee. A K-Nearest Neighbours (KNN) algorithm was used in MATLAB to create a classification model that was used to predict the aromas of beer and coffee, and was validated using a 10-fold cross validation (k-fold)... A 98 percent classification accuracy was achieved in the testing process."

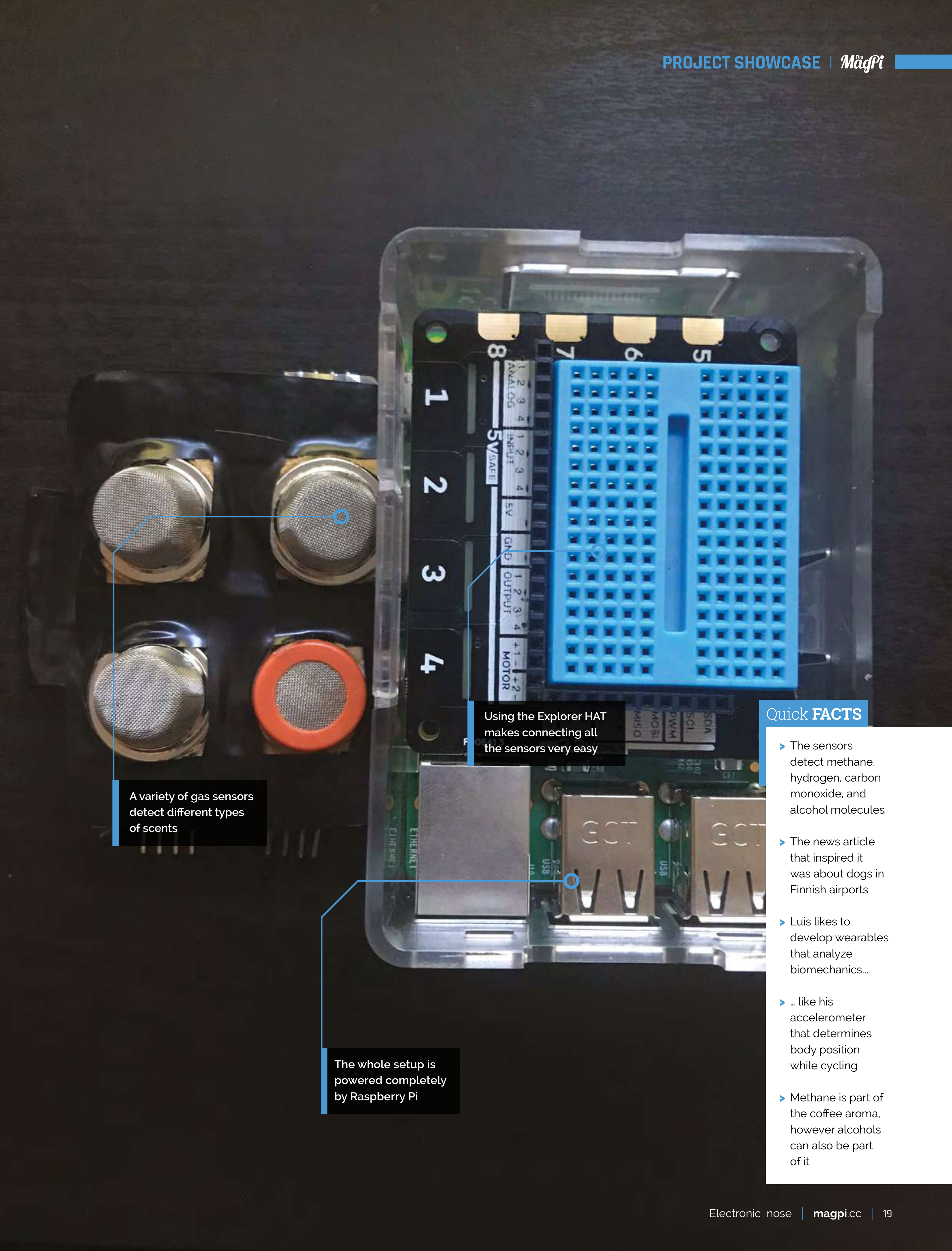
Smell test

With only four types of gas sensors, extensive testing and training of the model was required.

"A training data set was created by taking measurements of air, beer, and coffee independently." Luis explains. "Each sample was taken, on average, for 15 minutes at one second intervals, producing over 900 sample readings per test and the data was exported into CSV files. For classification purposes, an additional column was manually added to label the sample (i.e., coffee, beer, air). The three datasets were imported and combined in MATLAB. This data was used to create a k-nearest neighbour model, k was selected to be 5, this was determined by trial and error. A 10-fold cross-validation was used to validate the



The principal component analysis gives a more accurate reading of what data is being collected



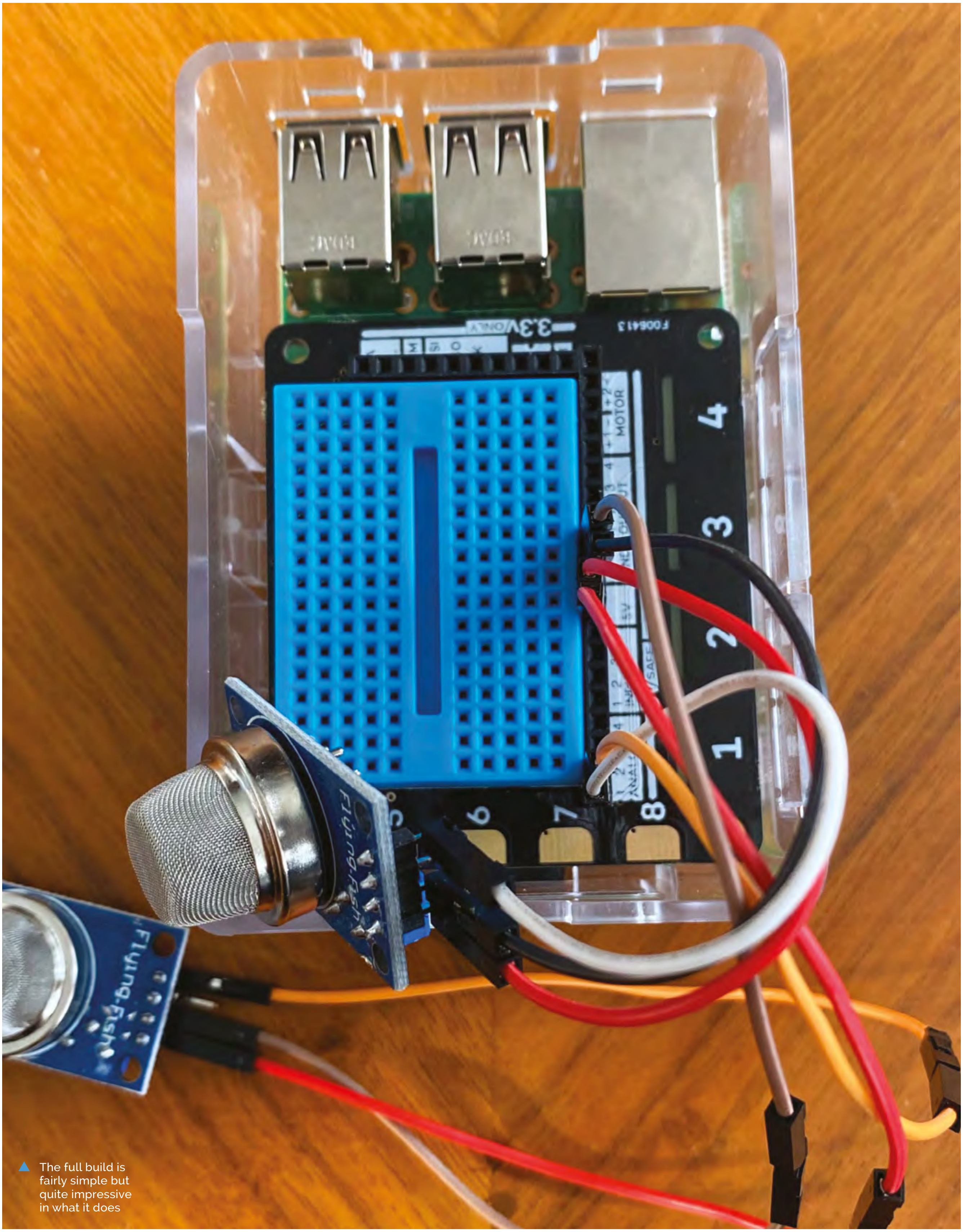
A variety of gas sensors detect different types of scents

Using the Explorer HAT makes connecting all the sensors very easy

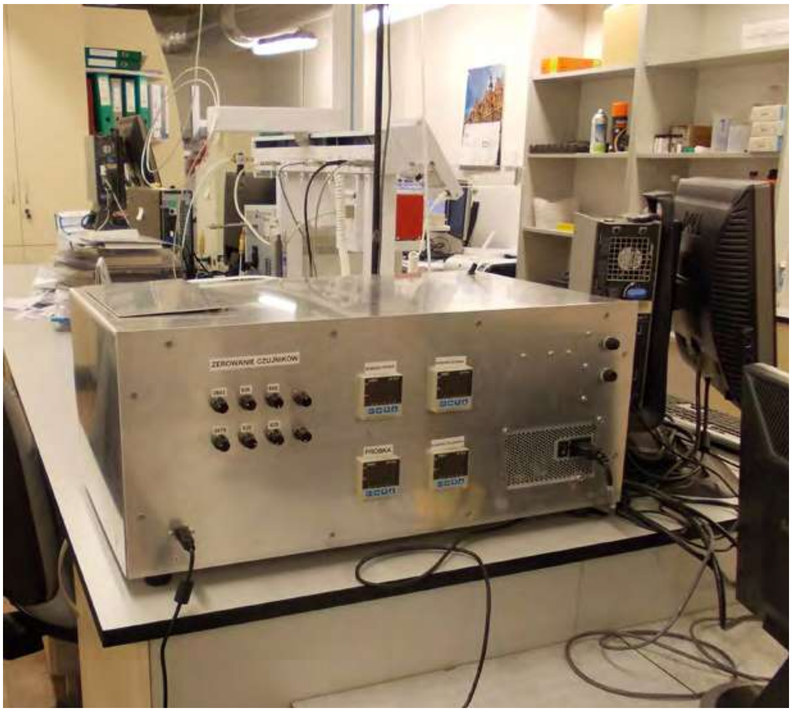
The whole setup is powered completely by Raspberry Pi

Quick FACTS

- The sensors detect methane, hydrogen, carbon monoxide, and alcohol molecules
- The news article that inspired it was about dogs in Finnish airports
- Luis likes to develop wearables that analyze biomechanics...
- ... like his accelerometer that determines body position while cycling
- Methane is part of the coffee aroma, however alcohols can also be part of it



The full build is fairly simple but quite impressive in what it does



▲ Professional electronic noses are used in medicine and engineering sectors and can cost thousands

Credit: *CC BY-SA 4.0: LukaszKatlewa*

model, and a Principal Component Analysis (PCA) was used as an exploratory technique to verify the model and the results, similar to the work shown in past research.

"A test dataset was gathered by taking 17 new samples of two-minute readings at one second intervals to assess the classification model. Each sample was independent of each other (only air, beer, or coffee was measured at a time), and they were manually labelled accordingly, resulting in over 2500 measurements. This data was imported, combined, and randomly rearranged in MATLAB. Using the classification model created from the training dataset, the testing data was classified and the results from the classification model represent 97.7% accuracy."

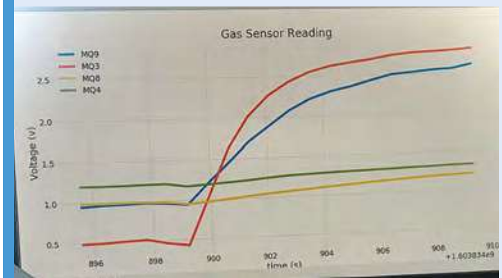
> I quickly realized how easy, efficient, and capable Raspberry Pi boards are

A near 98% accuracy is extremely impressive for the three test subjects, and it's all done on a Raspberry Pi 3.

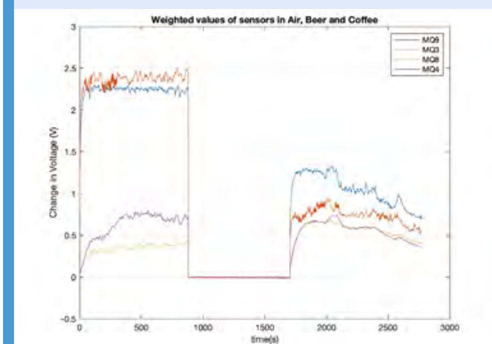
"Raspberry Pi was introduced to me in the fall of 2020 during one of my university courses," Luis said. "I quickly realized how easy, efficient, and capable Raspberry Pi boards are."

It's a cool, working concept, so we hope to see more like it in the future.

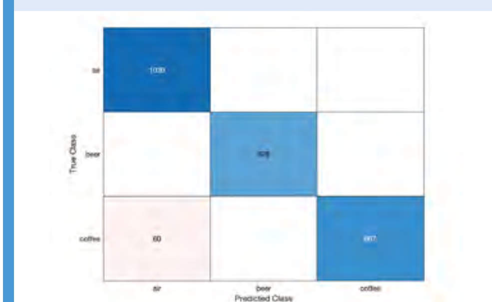
Sniffing electronically



01 The sensors are calibrated when it turns on – this is done by having the Python script run on start up, and holding the sensors away from anything you'd like to whiff. Once the sensors are calibrated to neutral, a sensor reading will be displayed.



02 Placing the e-nose on the sample will begin a reading using the gas sensors, which will be automatically logged in a CSV file for you to look at later.



03 The CSV file can be processed using MATLAB, and there are a couple of post-process scripts that Luis have created to output at confusion matrix with classifications, along with PCA and the raw data.

Pico Light Arcade

This inexpensive build was created in a single evening and it's set to provide hours of fun, as **David Crookes** discovers

MAKER

Thomas Roth

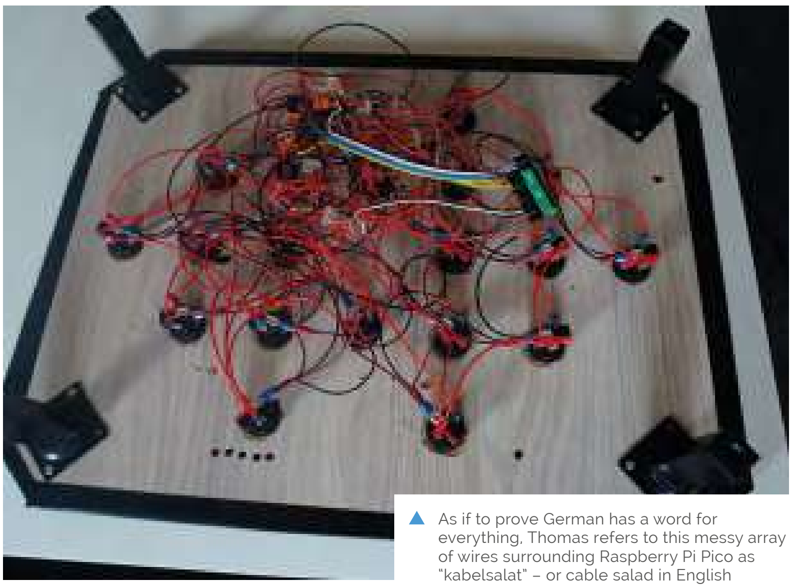
Thomas Roth is an IT security researcher who lives in Germany and creates videos about hardware hacking and reverse-engineering.

magpi.cc/picolight

A trip to a seaside arcade is rarely complete unless you've grabbed hold of a soft rubber mallet and exerted huge amounts of energy on a satisfying game of Whac-A-Mole. Known as Mogura Taiji in Japan, where the game originated, it relies on speedy reactions to hit as many small plastic moles as possible – and when Thomas Roth had a go, he decided he had to create a version for himself.

"It was a small arcade and my friend and I had so much fun with a game like this, I just knew I wanted to have one for our parties," he says. In his case, the 'moles' he was bashing were a set of LED lights and Thomas reckoned it would be relatively easy to replicate. "I spent zero time planning it," he continues. "I came back from the arcade, ordered the arcade buttons and, once they arrived, I drilled 20 holes into a board I had lying around."

The holes were arranged so that each player would have one button immediately in front of them, followed by a row of three and then a further five – making for nine each. Two buttons were placed in the centre: one to start the game and the other to select a playing mode. "I screwed in the buttons and wired them up in a 3×7 matrix," Thomas says. "There was no soldering involved."



As if to prove German has a word for everything, Thomas refers to this messy array of wires surrounding Raspberry Pi Pico as "kabelsalat" – or cable salad in English

Keeping score

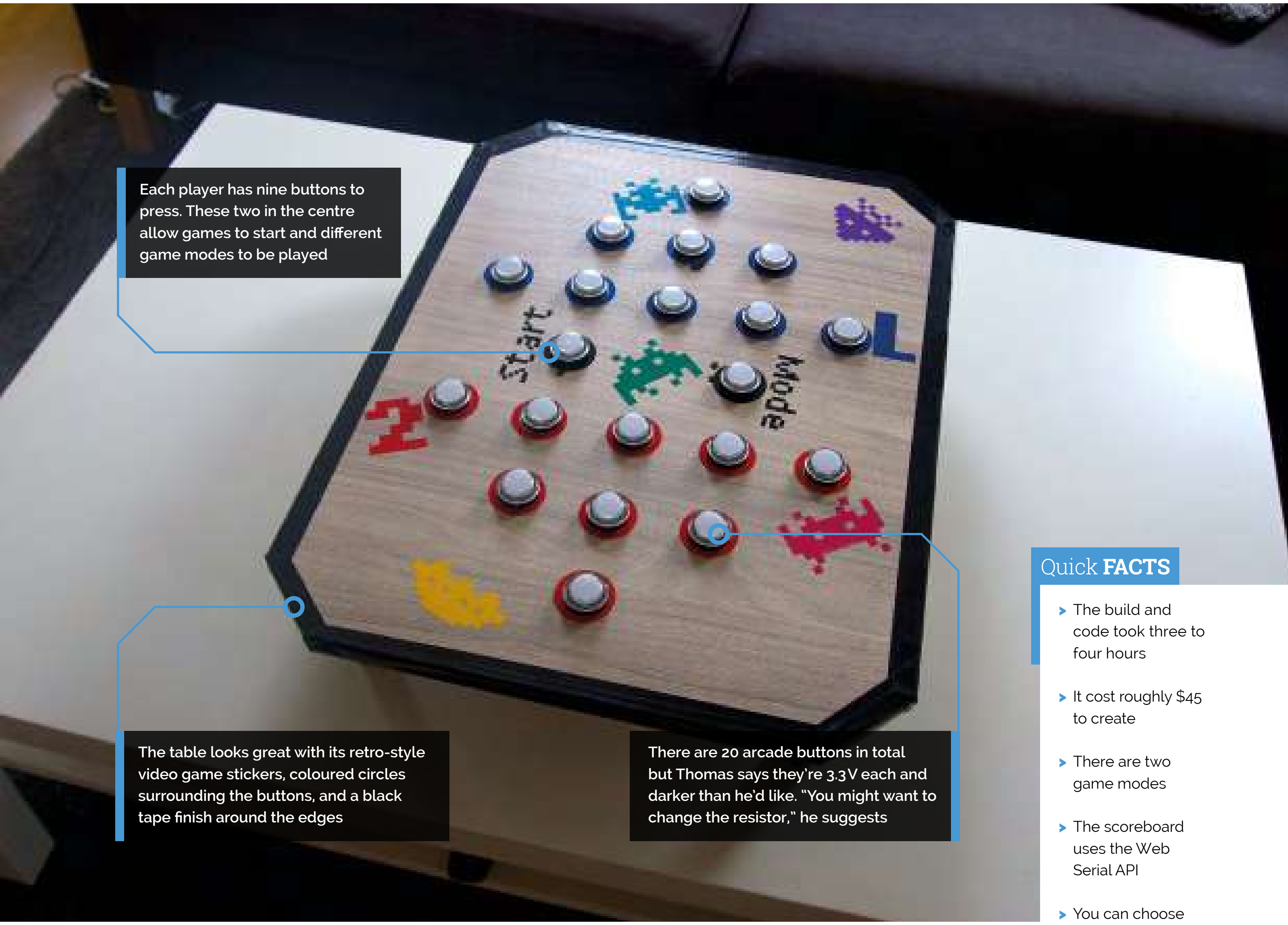
Once done, Thomas began to program the game and he reckoned a Raspberry Pi Pico microcontroller would be more than capable of running it. "I really like the simplicity of Raspberry Pi Pico," he says. "Firmware updates are drag-and-drop, it's super-easy to code for, it's really cheap, and it boots up immediately, too."

At first, he flexed his C programming skills to code what he describes as "a high-speed Whac-A-Mole" with the goal being to hit any buttons that light up as fast as possible. "Whoever manages to press the most buttons in 45 seconds wins," he smiles, adding that he then wanted a way of displaying the score.

"I didn't have a nice display lying around so I built a Web Serial-based website that also shows a scoreboard," he says. "This is completely optional, though, and the table will run happily without it."

Mind games

Once the rules of Whac-A-Mole were established and coded, he then moved on to create a second mode: replicating a game of Simon. "The table shows a sequence and both players have to remember and repeat it. The sequence gets longer



Each player has nine buttons to press. These two in the centre allow games to start and different game modes to be played

The table looks great with its retro-style video game stickers, coloured circles surrounding the buttons, and a black tape finish around the edges

There are 20 arcade buttons in total but Thomas says they're 3.3V each and darker than he'd like. "You might want to change the resistor," he suggests

Quick FACTS

- The build and code took three to four hours
- It cost roughly $45 to create
- There are two game modes
- The scoreboard uses the Web Serial API
- You can choose your own button layout

> "There was no soldering involved"

and longer, and whoever first presses a wrong button loses."

He's not ready to stop there, though. "I would love to have more game modes and I'd also love to see someone hook up a seven-segment display to show the score," he reveals. "I'm also designing a PCB to make the underside a bit cleaner – as it is, there are 20 buttons with four wires to each and it looks messy."

Even so, it also looks like a heap of fun. "It certainly is, and it should be a great project for anyone to do with kids," Thomas adds. "In fact, two friends of mine have already decided to build their own with their children, but adults will have a fantastic time with it too."

A laptop is being used to display the score, but it's only acting as a screen. Thomas would prefer to use a seven-segment display, but didn't have any to hand

Warning!

This project requires use of a drill. Be careful when using electric tools.
magpi.cc/drillsafety

Notes are fed to the tablet screen so everyone can follow along

A Raspberry Pi sends out all the info to the various tablets

A projector was experimented with, but it was too bright in the park

Community Jams

MAKER

Bob Steele

A retired software developer currently living in Portland. He writes software for the community jams and plays bass guitar with them.

communityjams.org

Bringing the jam sessions outside thanks to a Raspberry Pi broadcasting notes and lyrics. **Rob Zwetsloot** grabs a seat and listens

While reading *The MagPi*, you might think that a reference to a Jam would be for a Raspberry Jam. In this case you'd be wrong, it's about a community of musicians that like to play together.

Over the course of last year, the way they play together had to change. Mainly, they had to move outside and that caused some problems.

Who needs books of music when it gets sent to your tablet?

"Community Jams (**communityjams.org**) couldn't play music indoors using our web-based chords and lyrics application (**magpi.cc/musicapp**)," says Bob Steele, a retired programmer and current Community Jams member. "Outdoors, I put the app on a Raspberry Pi so we didn't need an internet connection. Raspberry Pi had a web server to deliver the application from Apache. Unfortunately, the projector was too weak in bright sunlight and was always subject to having its screen blow over in the wind. I figured out that if the app was delivered from [Apache] Tomcat, I could create a web socket protocol that would share the current song and position to many tablets. So now the musicians are outside, masked, separated by a good distance, and all able to share the same display on their personal tablets."

Early adopter

Although Bob has been programming and working with computers for several decades, he's mainly worked with software projects: "In the early 1980s

A very simple setup, which allows for the group to play together

Quick FACTS

- Bob used to work in Silicon Valley
- Jam sessions can be traced back to 1920s jazz
- Bassist Bob also has a bass guitar learning app on his site
- Community Jams spun off from Portland Casual Jams...
- ...which was created 14 years ago and has 3700 members

I designed or help design a number of internal hardware projects for the video industry.

"I heard about [Raspberry Pi] from the internet in the early days," he says. "I've ended up buying

> The musicians are happy that: 1) they get to play music in the park and 2) they know next to nothing about how it works

at least one of every model. Don't know why, but they are good platforms for small projects. I first used UNIX in 1975 and my personal machine has been Linux for many decades ... so Raspberry Pi seems like home. In fact, most of them never get connected to a keyboard, mouse, or display. SSH or VNC is all I usually need."

Can't beat it

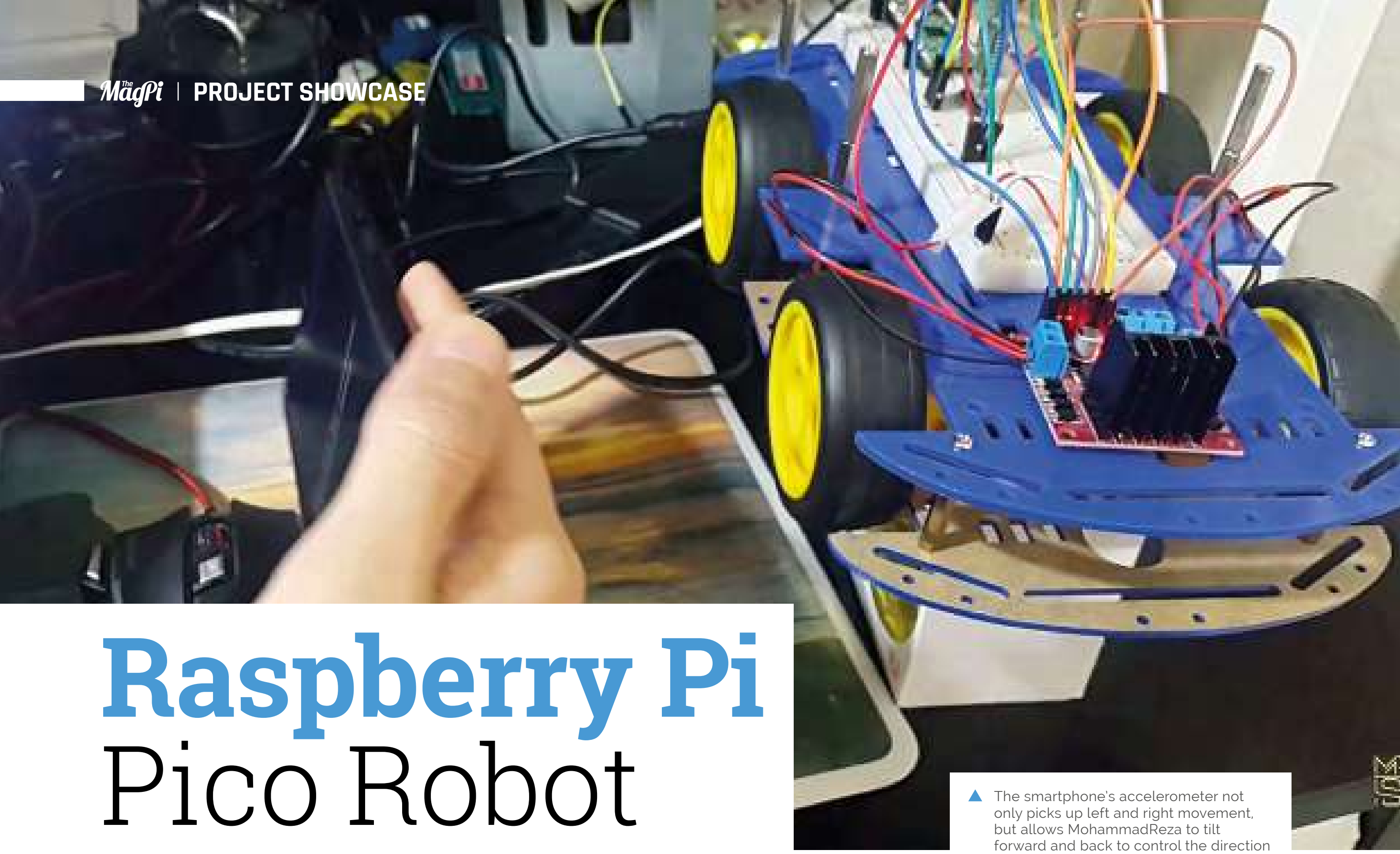
It's a simple project with a simple goal – to display music for musicians, and it works well in that regard according to Bob: "The musicians are happy that: 1) they get to play music in the park and 2) they know next to nothing about how it works."

The web app that Bob created, as seen on several screen sizes

The whole setup is just a Raspberry Pi 3 with Bob's software, an access point to connect to a local wireless LAN, and a battery to run it.

"I should have installed DNS instead of the fixed IP address," Bob ruminates. "The fixed IP address helps when Raspberry Pi is not in the park. It gets plugged in, attaches to the local WiFi, and updates itself using crontab jobs."

While its humble use as a way to get some friends to play music together is great, we feel the technology is simple yet powerful enough to be expanded to larger groups. Check out Bob's website, **bsteele.com**, for more about Community Jams and his other projects.



Raspberry Pi Pico Robot

▲ The smartphone's accelerometer not only picks up left and right movement, but allows MohammadReza to tilt forward and back to control the direction

Rather than use a standard controller, this robot is steered using an Android phone's accelerometer. **David Crookes** takes a look

MAKER

MohammadReza Sharifi

MohammadReza is a self-described geek, gamer, writer, cinema lover, YouTuber, and engineer. He lives in Khuzestan, Iran.

magpi.cc/picorobot

Robotics projects have long been popular among the Raspberry Pi community, but MohammadReza Sharifi's outstanding contribution definitely has a touch of va va voom about it. Powered by a Raspberry Pi Pico, his four-wheel drive car is controlled using the accelerometer in an Android smartphone. "The idea suddenly came to my mind," he tells us, "and it took me about three hours to build and program."

That itself is impressive, particularly given the work needed to get things moving. It all began two months ago when MohammadReza was browsing Instagram. "I saw a promotional video about a toy in which a coloured ball was being controlled by mobile gestures," he says. "I thought to myself that controlling a robot using mobile gestures could also be an interesting project."

From that point on, he quickly looked to gather the equipment needed. To start with, he bought an inexpensive plastic robot car chassis kit which came complete with four yellow wheels and four DC gear motors. He then added a L298 motor driver and an HC05 Bluetooth module.

Speeding ahead

"I decided to connect these to a Raspberry Pi Pico microcontroller," he says. "I'd seen that a small number of robotics projects had been implemented using Raspberry Pi Pico, and I wanted to experience the challenge of working with the microcontroller."

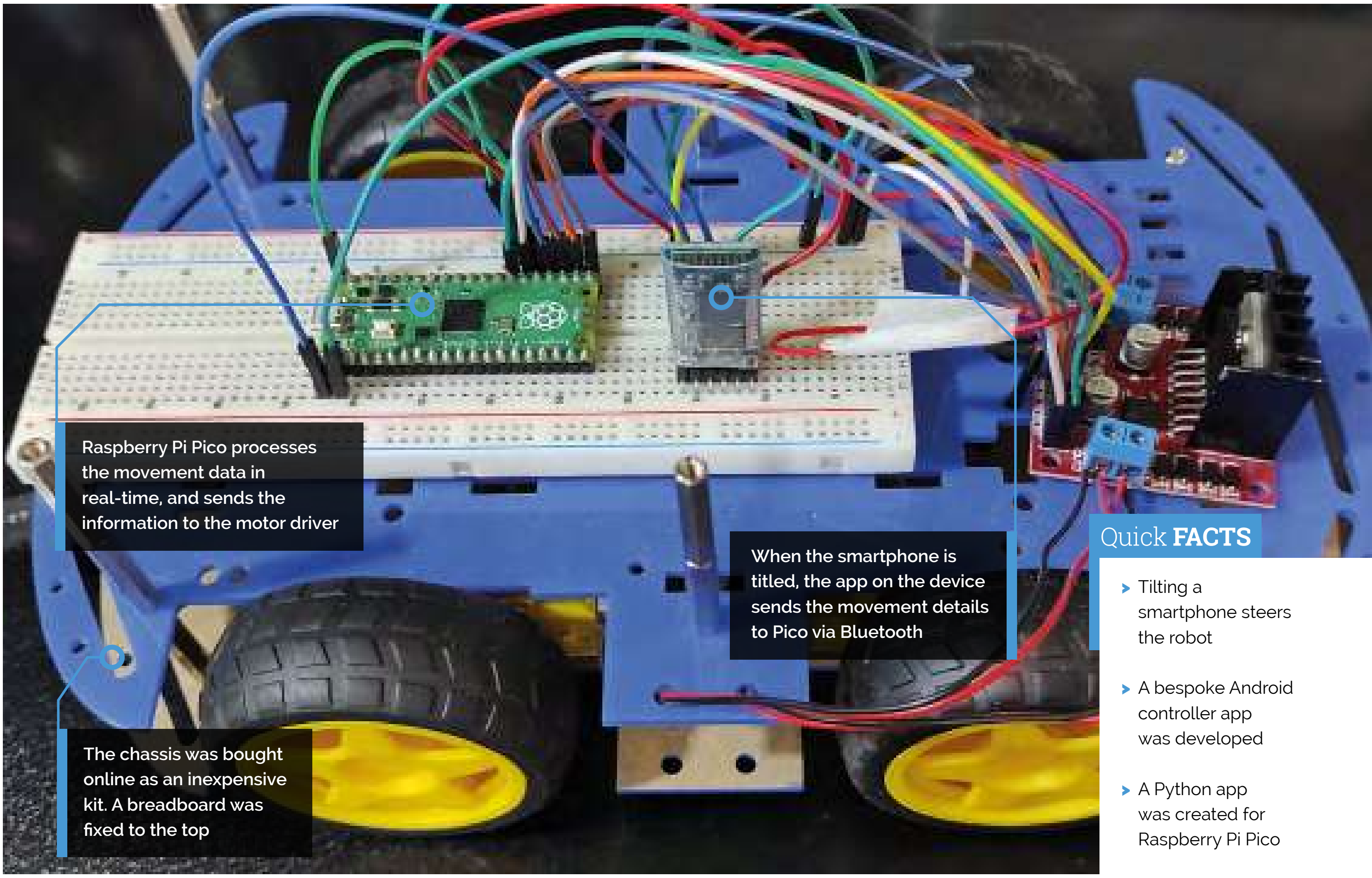
To ensure that the robot could make use of a smartphone's accelerometer, MohammadReza needed to develop a custom app. He did so using the MIT App Inventor platform (**appinventor.mit.edu**) – a blocks-based visual programming environment that makes it relatively easy to create software for mobiles.

> The biggest challenge was definitely building the mobile app

"The main advantage of this platform is that you don't have to deal with coding challenges because it lets you focus more on the main concept of a project," MohammadReza explains. That said, it wasn't totally straightforward, as you can perhaps imagine.

Twists and turns

The idea was to create an app that would send data from the accelerometer to Raspberry Pi Pico in real-time via a Bluetooth connection. That way,



Raspberry Pi Pico processes the movement data in real-time, and sends the information to the motor driver

When the smartphone is titled, the app on the device sends the movement details to Pico via Bluetooth

The chassis was bought online as an inexpensive kit. A breadboard was fixed to the top

Quick FACTS

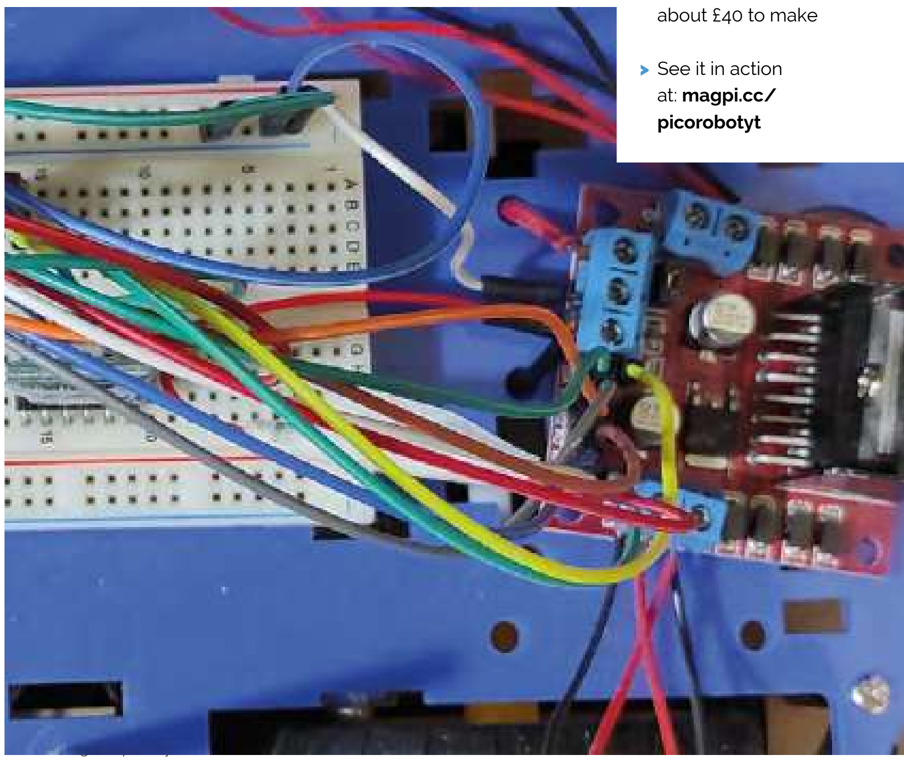
- Tilting a smartphone steers the robot
- A bespoke Android controller app was developed
- A Python app was created for Raspberry Pi Pico
- The project cost about £40 to make
- See it in action at: **magpi.cc/picorobotyt**

MohammadReza could tilt his smartphone left, for instance, and send information telling the robot to move left.

This is what took up much of the development time. "The biggest challenge was definitely building the mobile app," he reveals. "It needed to identify the co-ordinate axes and identify the position of the accelerometer sensor in the mobile. It also needed to perform trial and error to find the amount of angular acceleration and the magnitude of acceleration."

Eventually, however, it began to work near-flawlessly. He's now considering creating a version for iPhone, but although the MIT App Inventor web application is free, open-source, and allows apps to be made for iOS and Android, he's decided to use a different system for the Apple device.

"I will create an iOS app for the robot using Kivy," he says, referring to the free and open-source Python framework and relishing a fresh challenge. His priority first and foremost this time around, he adds, was just getting it to work with his Xiaomi Redmi Note 9 Pro Android smartphone and he's more than happy to see it going down well with other makers online. "It makes me proud that people are interested in my project," he beams.



Oasis-grow

Growing fruit and vegetables can be so frustrating, but a Raspberry Pi-based smart agriculture project helps take the uncertainty out of cultivation. **Rosie Hattersley** reports

MAKER

Mike Lee and Aviel Stein

Oasis-X co-founders Mike and Aviel met at a maker event in Pennsylvania where self-taught coder and entrepreneur Mike was showing off several Raspberry Pi projects.

oasis-x.io

"Plants, mushrooms, and other cultivated organisms don't respond in linear ways," even when provided with supposedly 'ideal' growing conditions, much to the frustration and disappointment of those keen to cultivate their own food, observes 25-year-old entrepreneur and technology fan Mike Lee. Such nonconformity "is a mindset that many programmers and engineers find difficult to accept," says Mike, explaining why he and his PhD student co-founder Aviel Stein set about improving the odds of a successful harvest. Their response, Oasis-grow, is a 'smart agriculture engine' that runs on Raspberry Pi and collects detailed information about whatever the user cares to grow in a bid for more predictable results.

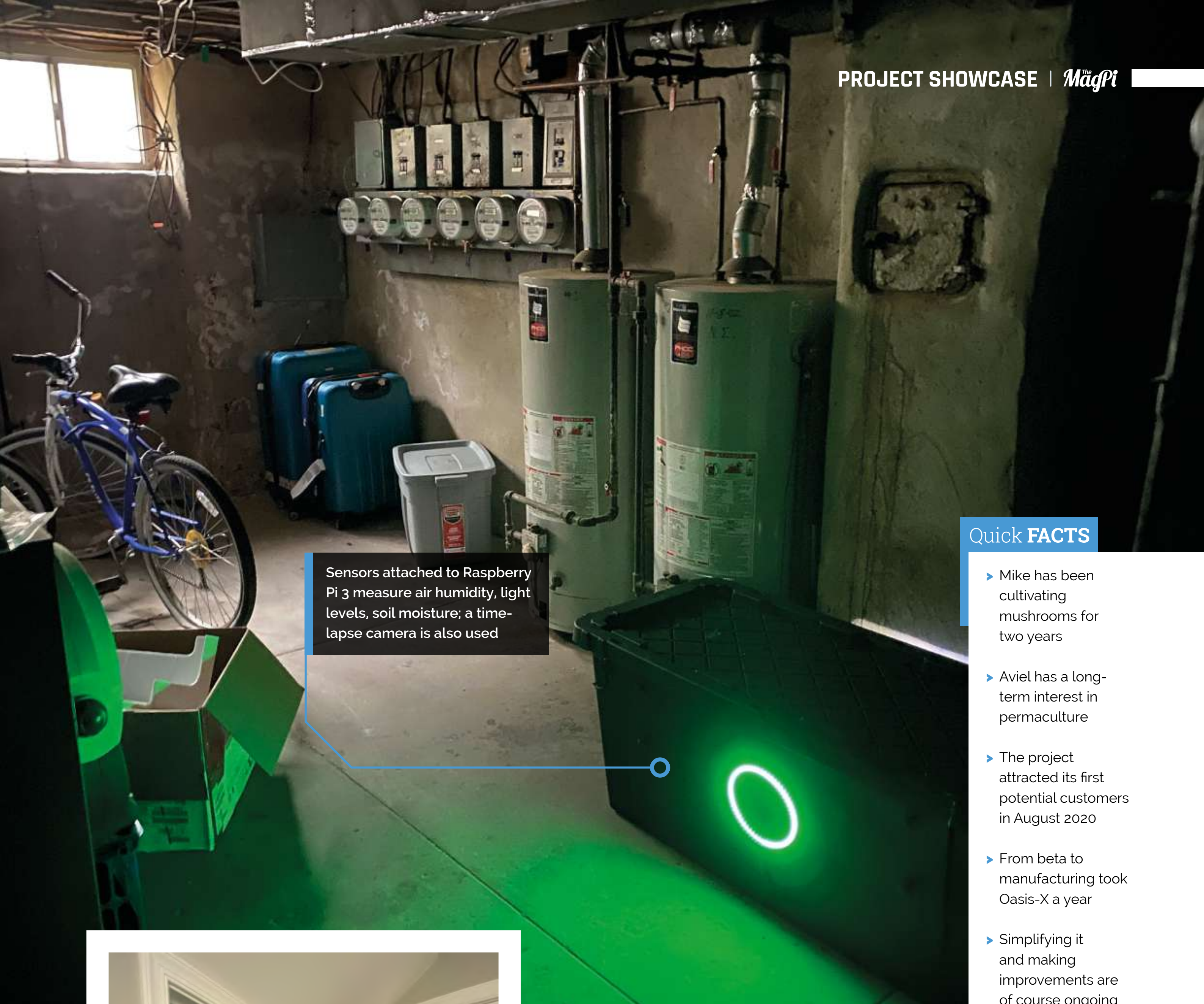
The temperature-controlled incubation tent has sensors to monitor humidity, airflow, and light levels

The system provides visual and sensor data, plus controls for heat, humidity, airflow, light, and watering cycles. The selling point is a system that "keeps your plants, mushrooms, and Petri dishes happy, and remotely monitors the grow space so you can identify and address problems before they get serious."

Make room for mushrooms

Mike and Aviel met at the 2019 Pennsylvania Maker Faire where, among other projects of his own devising, Mike presented his first Raspberry Pi project: a dashboard camera with buttons and a remote, back-window LED screen for signalling other cars. "It was very cool but perhaps a bit too dangerous for the road (the other cars did not react … well)," he admits. Nonetheless, the pair decided to work together, and quickly formed Oasis-X (**magpi.cc/oasisx**) having identified a market for a smart agriculture monitoring tool.

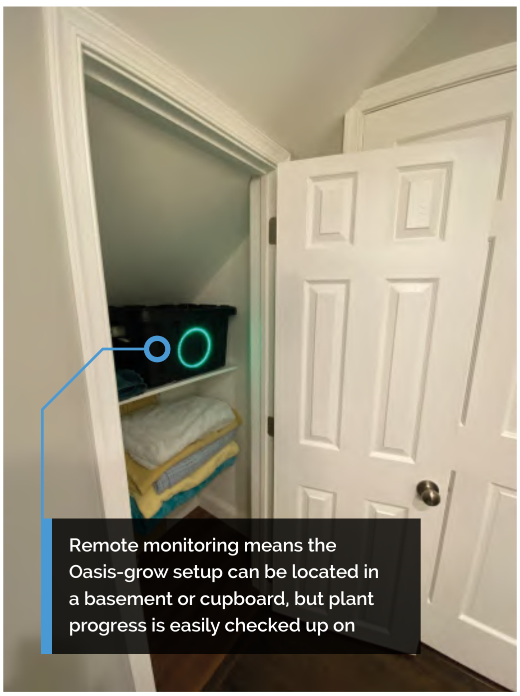
Aware that DIY plant monitoring and sensor kits were already available, Mike and Aviel needed to be sure they were offering something that fulfilled the scalability promise it would need to be of interest



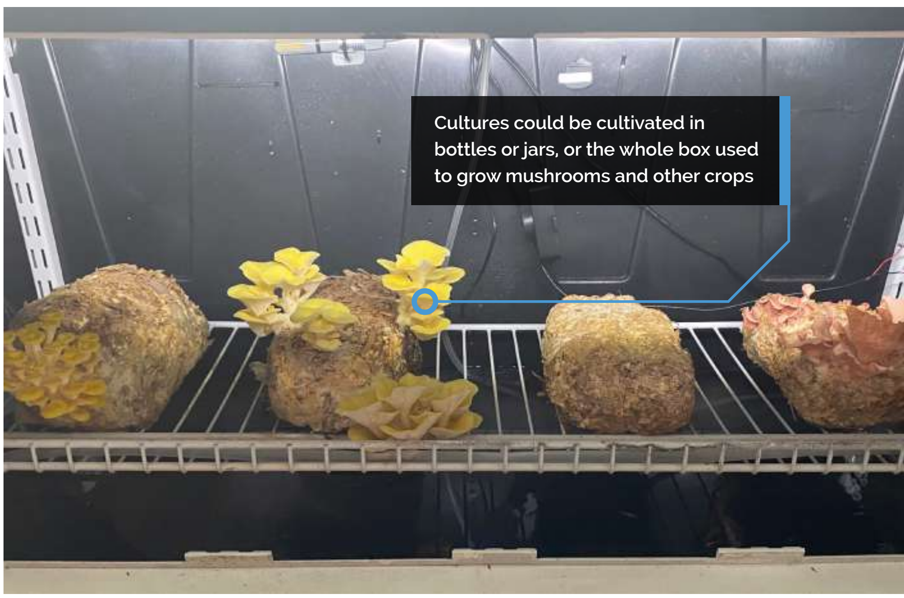
Sensors attached to Raspberry Pi 3 measure air humidity, light levels, soil moisture; a time-lapse camera is also used

Quick FACTS

- Mike has been cultivating mushrooms for two years
- Aviel has a long-term interest in permaculture
- The project attracted its first potential customers in August 2020
- From beta to manufacturing took Oasis-X a year
- Simplifying it and making improvements are of course ongoing



Remote monitoring means the Oasis-grow setup can be located in a basement or cupboard, but plant progress is easily checked up on



Cultures could be cultivated in bottles or jars, or the whole box used to grow mushrooms and other crops

Warning!
Electrical Safety

Please be careful when working with electrical projects around the home. Especially if they involve mains electricity.

magpi.cc/electricalsafety

Time-lapse images can be taken of the contents in their Petri dish habitat

to agriculture, and merit a commercial launch. Mike explains it was important for Oasis-X to take an "open-source first approach" to developing agriculture technology. "Technology vendor lock-in is such a big issue for farmers," he continues.

> Oasis-X is also keen to make the concept available for hobbyists and home horticulturists, providing source code and hardware architecture details via GitHub

"Our systems must collect and store lots of different information while communicating with the network and managing multiple independent processes, [so] a full-fledged operating system is close to required."

Ploughing ahead

Oasis-grow's makers took inspiration from older projects, and chose Raspberry Pi because of its multitasking capabilities and a file system that they found made organising and retrieving complex data a much simpler task. Most of the hardware they use for prototyping was bought from the usual online retailers, with Raspberry Pi running Bash alongside Raspberry Pi OS and Python. Their web interface was created in Python and uses AWS (Amazon Web Services).

"We've spent the last year nailing down the core functionality and are in the process of making it faster, more modular, and easier to use," Mike explains. The system went through more than 20 iterations and was developed in close consultation with farmers and home horticulturists. This process helped Mike and Aviel realise that the vision, sensing, and environmental helper modules were valuable individually as well as

▲ Impressive results from their initial home growing chambers

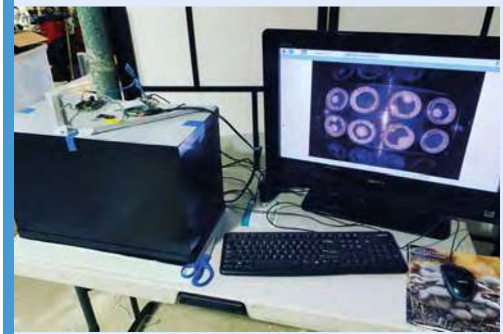
collectively. As a result, the pair are soon set to offer an open-source version of the GUI integrating machine learning, decision support, and smart notification prototypes.

As an agri-business system, Oasis-grow had to prove its worth and has undergone successful long-term, large-scale field tests in farms in the eastern US over the past year. These 'turn-key growing systems' have recently been distributed to the company's early backers, with a Kickstarter campaign fundraising towards its 'smaller helper modules' currently running. However, Oasis-X is also keen to make the concept available for hobbyists and home horticulturists, providing source code and hardware architecture details via GitHub (**magpi.cc/oasisgrowgit**).

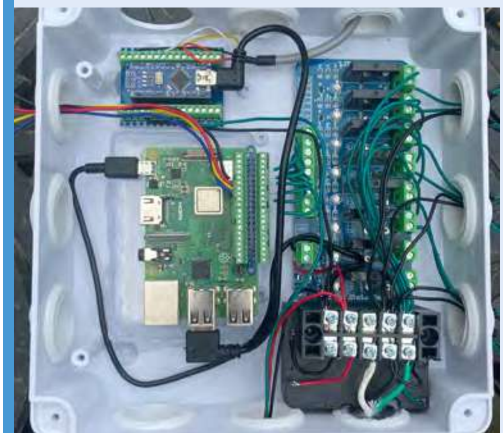
Grow your own

Raspberry Pi 3B, a USB camera, and a power supply are the basis of this project. Full setup instructions can be found at **magpi.cc/oasisgrowgit**.

01 Download the Oasis-grow code from the GitHub page (an installable operating image is in development, take a look at the GitHub instructions). Set up your incubation tent, or position the camera to monitor it.



02 To start data collection, turn on the Raspberry Pi camera, start Python, and run **main.py**. If successful, the image loader will run on bootup next time.



03 For more detailed monitoring, add a relay board for AC control, GPIO buttons for a physical interface, environmental sensors, and LED status displays, as per Oasis-grow's full configuration.

SUBSCRIBE TODAY FROM ONLY £5

SAVE UP TO 35%

Subscriber Benefits

- **FREE Delivery**
  Get it fast and for FREE
- **Exclusive Offers**
  Great gifts, offers, and discounts
- **Great Savings**
  Save up to 35% compared to stores

Rolling Monthly Subscription

- Low monthly cost (from £5)
- Cancel at any time
- Free delivery to your door
- Available worldwide

Subscribe for 12 Months

£55 (UK) £90 (USA)
£80 (EU) £90 (Rest of World)

Free Raspberry Pi Zero 2 W with 12 Month upfront subscription only (no Raspberry Pi Zero 2 W with Rolling Monthly Subscription)

Subscribe by phone: **01293 312193**

Subscribe online: **magpi.cc/subscribe**

Email: **magpi@subscriptionhelpline.co.uk**

Introducing Raspberry Pi Zero 2 W

The smallest Raspberry Pi now packs a quad-core processor and runs over five times faster. Raspberry Pi Zero is back in a big way. By **Lucy Hattersley**

Raspberry Pi Zero is one of the greatest ever Raspberry Pi computers. This tiny powerhouse has been given away free to all *The MagPi* magazine subscribers ever since its introduction in 2015.

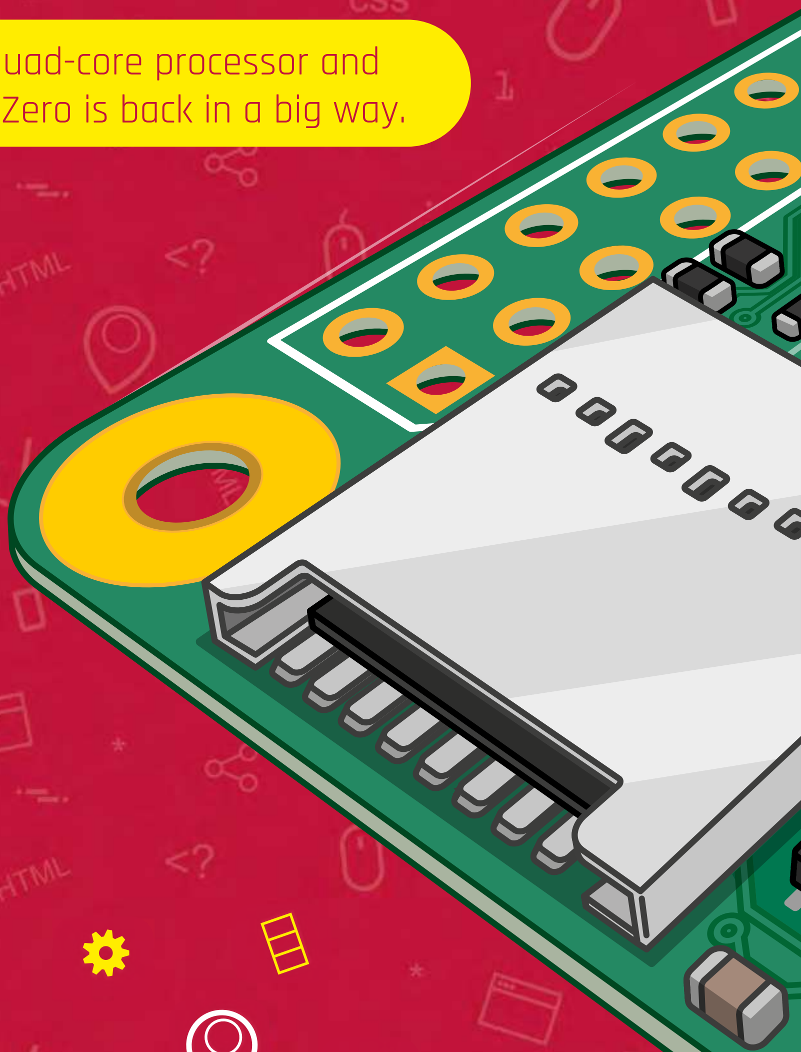
It may be diminutive in size, but Raspberry Pi Zero's reputation is enormous. This month we're delighted to introduce you all to Raspberry Pi Zero 2 W: it has a lot to live up to.

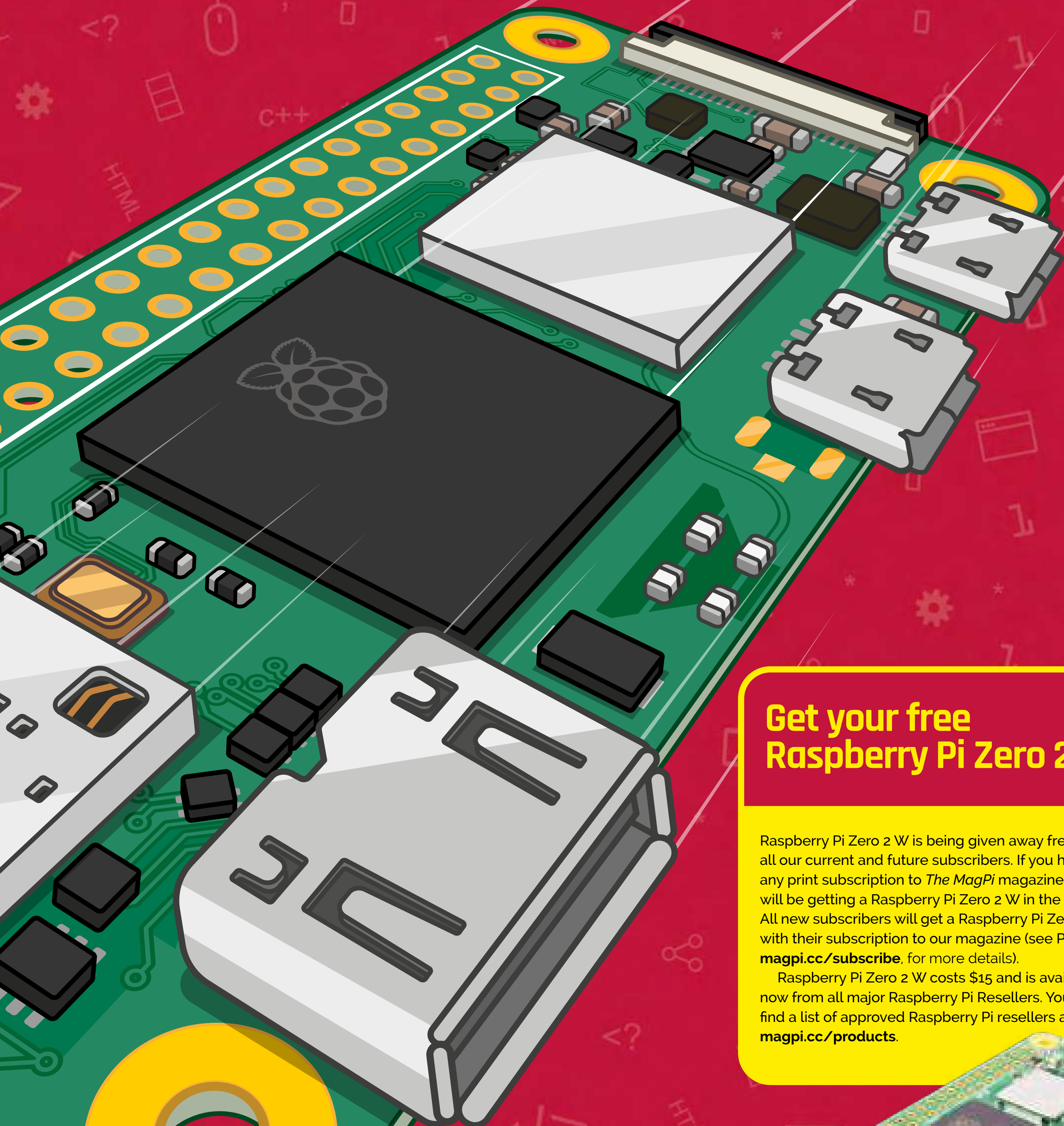
It comes in the exact same package you know and love. Only now, Zero 2 packs a more powerful Arm Cortex-A53 quad-core processor running at 1GHz.

Zero 2 sticks with the same form factor, so you can take a Raspberry Pi Zero out of your current project, and drop Raspberry Pi Zero 2 W straight in its place and immediately benefit from the improved speed. It's also compatible with the vast array of kits and innovative projects designed around the small board.

At the heart of Raspberry Pi Zero 2 W sits an all-new SiP (System-in-Package). An exciting new approach by Raspberry Pi, that combines the usual System-on-Chip with DRAM and enables a faster CPU to sit in the same form factor.

It's a new, and faster, day for Raspberry Pi Zero, yet one that keeps the important heritage intact. We can't wait to see what you make with it.





Get your free Raspberry Pi Zero 2 W

Raspberry Pi Zero 2 W is being given away free to all our current and future subscribers. If you have any print subscription to *The MagPi* magazine, you will be getting a Raspberry Pi Zero 2 W in the post. All new subscribers will get a Raspberry Pi Zero 2 W with their subscription to our magazine (see Page 32, **magpi.cc/subscribe**, for more details).

Raspberry Pi Zero 2 W costs $15 and is available now from all major Raspberry Pi Resellers. You can find a list of approved Raspberry Pi resellers at **magpi.cc/products**.

Meet Raspberry Pi Zero 2 W

Raspberry Pi Zero 2 W is designed to be instantly swappable with Raspberry Pi Zero in a current project, with an identical form factor and most components in the same place. Dig a little deeper, and there are a host of tweaks and improvements across the board.

Specifications

SOC:
Broadcom BCM2710A1 quad-core Arm Cortex-A53 (ARMv8-A) 64-bit @ 1GHz

GPU:
Broadcom VideoCore IV

RAM:
512MB DRAM

NETWORKING:
802.11 b/g/n wireless LAN, Bluetooth 4.2 (Bluetooth Low Energy, BLE)

GPIO:
HAT-compatible 40-pin GPIO header, unpopulated

STORAGE:
microSD

PORTS:
Micro SDCard slot, Mini HDMI, USB On-The-Go, micro USB power; composite video and reset pins (via solder test points), CSI camera connector

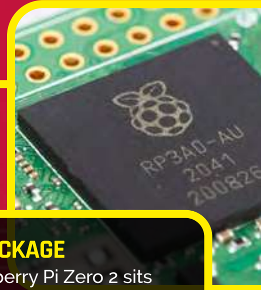
POWER CONSUMPTION:
1 W (idle in Raspberry Pi OS GUI); Max power, 3.8 W or 0.76 A (32-bit mode); 5.5 W or 1.1 A (64-bit) mode

PRICE:
$15

DIMENSIONS:
66 × 30.5 × 5 mm form-factor (compatible with Raspberry Pi Zero)

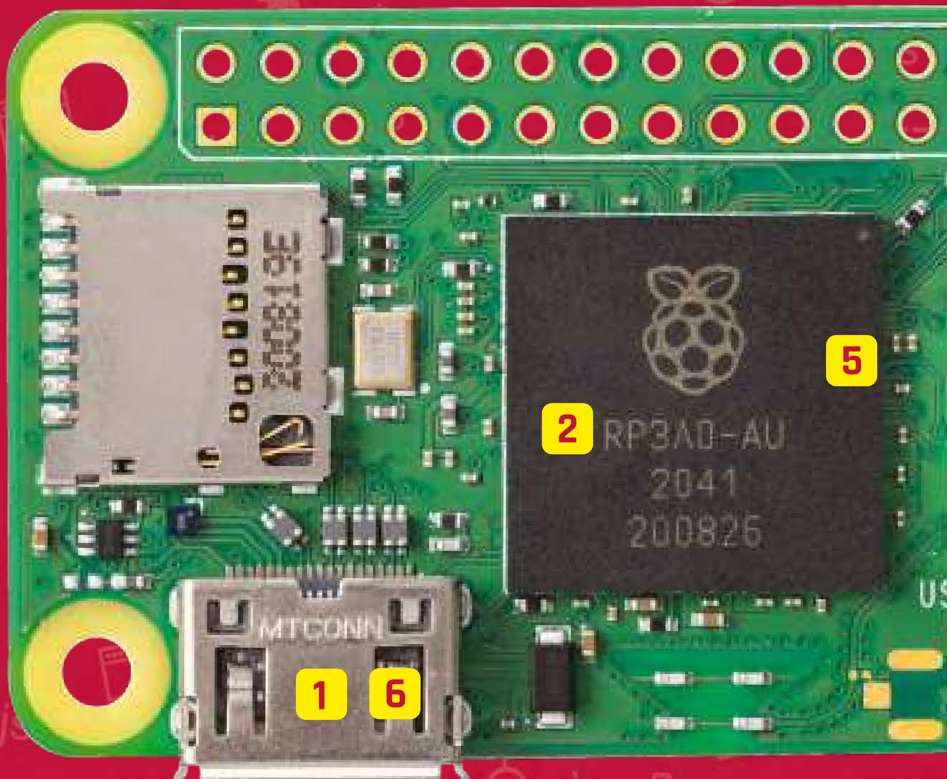
1 MINI-HDMI

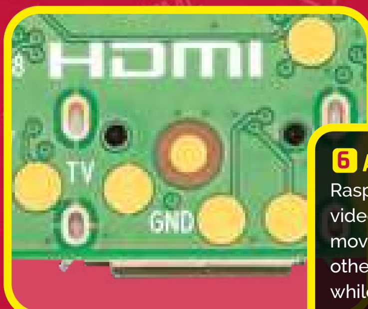
The Mini-HDMI port now has winged edging that enables easier insertion of the HDMI cable.



2 SYSTEM-IN-PACKAGE

At the heart of Raspberry Pi Zero 2 sits a System-in-Package (SiP) containing a Broadcom BCM2710A1 quad-core Arm Cortex-A53 System-on-Chip (SoC) with VideoCore IV and 512MB DRAM.



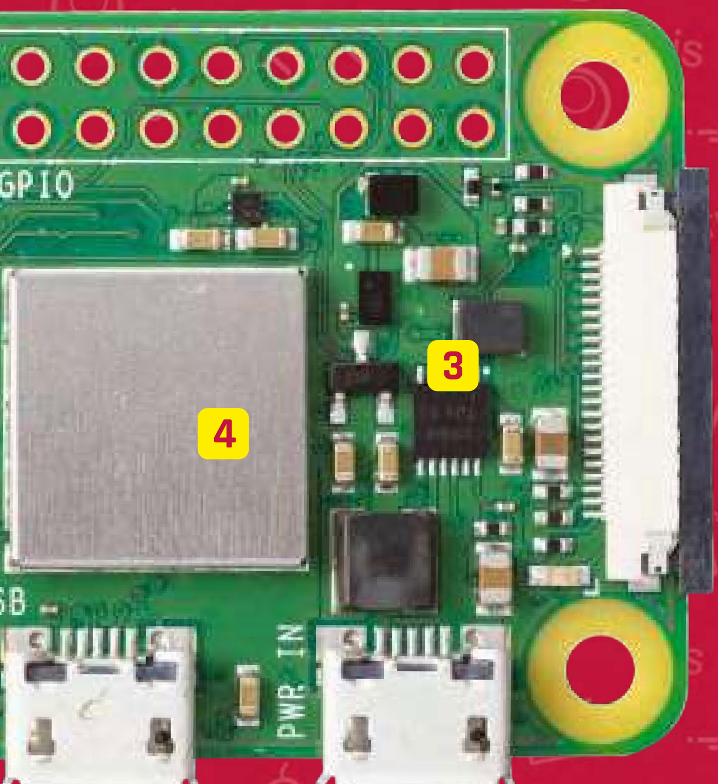


6 ANALOGUE VIDEO AND RESET

Raspberry Pi Zero 2 W supports analogue video and reset connections. The pins are moved to the rear of the board, alongside other test pins. In the bottom-right sits TV, while in the top-left, a Run pin can be used to reset and restart Zero 2.

3 NEW POWER CIRCUIT
The enhanced power circuitry from Raspberry Pi 3 has been shrunk down and fitted onto Raspberry Pi Zero 2 W's board.

4 WIRELESS LAN
The wireless LAN and Bluetooth is now housed inside a module enclosure. This can help with compliance, enabling Raspberry Pi Zero 2 W to have a smoother entry to highly regulated industrial environments.

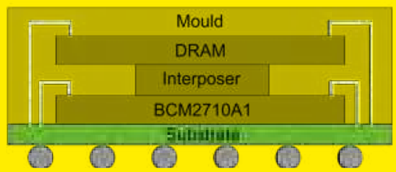


5 CAPACITORS INSIDE CPU
There are now 18 capacitors inside the SiP and fewer 0201 capacitors on the board. When the CPU demands a surge in power, the capacitors inside can ensure that power is there to meet demand.

Meet the System in Package

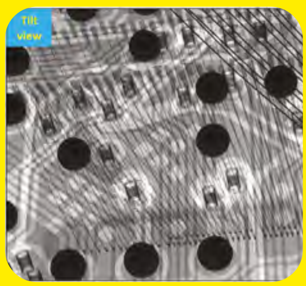
One interesting new aspect of Raspberry Pi Zero 2 W is the System-in-Package approach to the chipset design.

A Broadcom BC2710A1 SoC is placed on the substrate and 512MB of DRAM is placed above, with an interposer placed between the two (to act as a 'spacer'). This enables both chips to connect neatly to the substrate using gold wire. The whole thing is placed inside a mould. This is the black chip you see on the Raspberry Pi board.

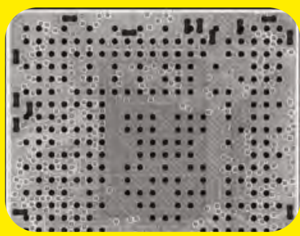


X-ray images

The picture is a view of a 3 mm × 2 mm section of the System-in-Package. The black circles are solder balls. The thick grey lines are tracks on the substrate. The thin black lines are gold wires, and you can see them bonding to the silicon (which is transparent to the X-ray machine). Here, we can see the 0201 (50 um × 25 um in size) capacitors inside the package.



Here is a top-down X-ray of the System-in-Package that reveals an Easter egg! Spot the Raspberry Pi logo shape in the ball map. This is possible because, while the connections on the outside are signals, the ones in the middle are ground/power and can be arranged into any pattern.



Eben Upton and Simon Martin on Raspberry Pi Zero 2 W

Raspberry Pi founder and CEO Eben Upton, and Principal Hardware Engineer **Simon Martin**, walk us through the design of the new Zero 2

We caught up with Simon Martin, Principal Hardware Engineer at Raspberry Pi and Eben Upton, founder and CEO of Raspberry Pi.

Simon: "Raspberry Pi Zero 2 W is all about how much power you can pack into such a tiny space. It's about just how much can we get from such a small form factor.

"Keeping that same form factor is important. An original Raspberry Pi Zero can be removed from a project and Raspberry Pi Zero 2 W can be plugged into its place. Any application that was there beforehand will get a boost from the processor performance.

"Instead of using the single-core processor that's in the Raspberry Pi Zero, we've got a quad-core Cortex-A53 processor, which is similar silicon to the original Raspberry Pi 3 when that was announced."

Eben: "There's this whole question of 'can we do it again?' You can use Moore's Law in two ways: you can take an amount of money and fill [Raspberry Pi] with more and more computing power, or you can take the current amount of computing power and deliver that at a lower and lower cost. And that's what we did with Raspberry Pi 1 – we took a ten-year-old PC's processing power and delivered it at around $30. And then we launched Raspberry Pi 2 in 2015 and, lo and behold, we've done the thing that everyone else does: we've picked a price point and filled it out with six times as much computing power using Moore's Law.

"Raspberry Pi Zero is really the result of going 'Aha! We should do the same thing as we did before.' We should take a Raspberry Pi 1's worth of computing power and we should use Moore's Law to squeeze the price down.

"Every part on that board pays for itself. It's a single-sided board, and it only has reflowable components; it doesn't have any through-hole components. So it can all be manufactured simply and robotically.

"So, coming up to the sixth anniversary, and how to use everything we've learned from those five years, there's this whole question of 'can we do it again?' And we're delivering most of Raspberry Pi's 3 power for not much more than half the price of a Raspberry Pi 3.

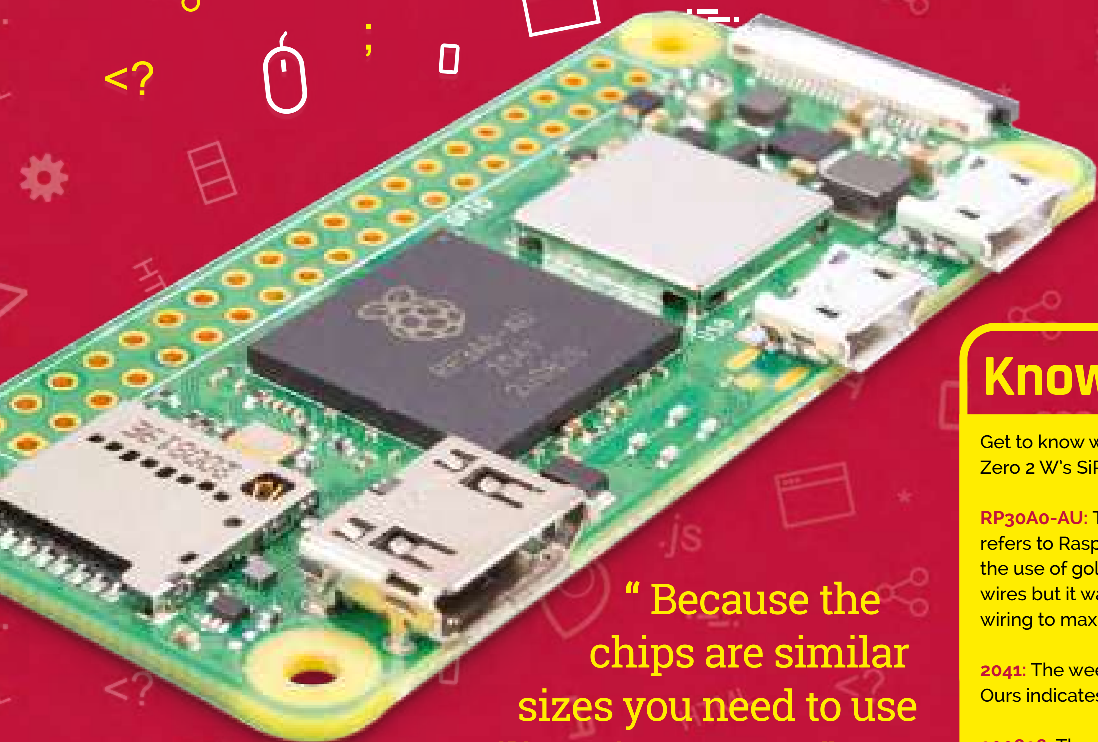
"The reality is that earlier this year people found out that Raspberry Pi knows how to make their own silicon [see, Raspberry Pi RP2040 in Raspberry Pi Pico, **magpi.cc/102** – Ed]. Raspberry Pi Zero 2 W is not our own silicon, but we package silicon in unusual ways

Inside the package

Simon: "It's also known as a 'stacked package'. It's the concept of putting more than one monolithic piece of silicon into a package to make a full system and package device. There's a BCM2710A1 SoC (system on a chip) in there and half a gigabyte of DRAM. These are both gold-wire stitched to the substrate. That is then encapsulated to make the package. And so, by doing this, we can get two chips into the space of one.

"Because the chips are similar sizes you need to use a silicon interposer, or spacer, amongst the centre of the processor. Then, when you put the memory on top, it doesn't squash the gold wires that connect the processor to the substrate. The memory is very sensitive, so there's less desire to put anything on top of the memory, which is why the SoC goes on the bottom and the memory on top.

"The A53 was the highest performance chip that we've ever used that used wire bonding in



> " Because the chips are similar sizes you need to use a silicon interposer "

Know the numbers

Get to know what the numbers on the Raspberry Pi Zero 2 W's SiP mean:

RP30A0-AU: This is the SiP design number. The A0 refers to Raspberry Pi Zero, and the AU indicates the use of gold wiring. The first models used copper wires but it was changed to more expensive gold wiring to maximise long-term reliability.

2041: The week of manufacture (read right to left). Ours indicates the 41'st week in 2020.

200826: The date the batch was ordered (read right to left). In this case, 26 August 2020.

the package, and Raspberry Pi Zero 2 W displays a considerable uptick in performance. Single-threaded performance is approximately 40% faster than a normal Raspberry Pi Zero. Obviously, this is quite cool, but if you do a benchmark of the single-core on a Raspberry Pi Zero and compare it to the quad-core of Raspberry Pi Zero 2 W, you get more than five times the performance. Real time performance is around three times the speed. A Raspberry Pi Zero takes around 90 seconds to boot into the GUI, whereas Raspberry Pi Zero 2 W takes around 30 seconds.

"It's a small board, and if you are willing to help out by putting a heatsink on it, or if you put it in some sort of metal case where the heat can be drawn away from the chip, then it is possible to add voltage to the device so you can run it faster." A Raspberry Pi Zero 2 W with a cooling solution can generally sustain 1.2GHz performance."

Eben: "We use thick copper inside the board. So, effectively, we dissipate the heat throughout the board. So if you look at Raspberry Pi Zero 2 W with a thermal camera, you'll see that it gets hot throughout the board. That's because we're using copper to move heat away from the CPU.

"The can [Wireless Lan enclosure] is a forward-looking thing to the days where people want to design this into things. We've got to be upfront: there's not a huge amount of stock for the first year because 'hey, there's a global semiconductor shortage.' Looking ahead, this is a product that's going to be around for a long time, and we hope people will put it into product designs for OEMs."

Raspberry Pi Zero 2 W Starter Projects

Here are some great ideas for things to make with your Raspberry Pi Zero 2 W

Handheld console

There are many retro gaming projects for Raspberry Pi, and Raspberry Pi Zero lends itself extremely well to small, handheld consoles. Attach a screen, battery, consoles, and a 3D-printed case, and you're good to play handheld classics.

There are some handy kits you can get to help build your handheld console, like this RetroFlag GPi Case.

> magpi.cc/retroflaggpi

Handheld camera

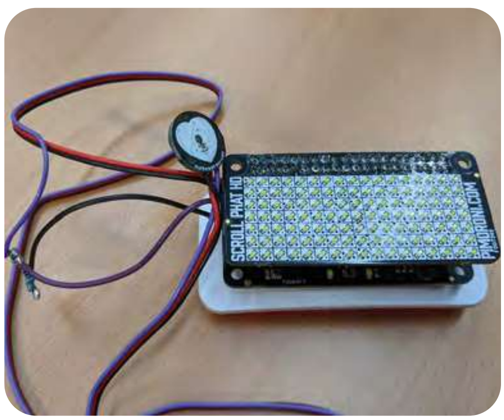
Raspberry Pi Zero 2 W has the CSI (camera serial interface) port found on most Raspberry Pi computers, enabling you to connect a Raspberry Pi Camera Module to the device. And the official case has a cover with a pin-hole space for the camera. Many makers have taken things further with 3D-printed cases and LCD screens. Take a look at PolaPi-Zero for inspiration, and a guide to make your own Raspberry Pi Zero 2 W camera.

> magpi.cc/polapi

Heart monitor

Practice checking on your ticker with this simple heart rate monitor using the Enviro pHAT from Pimoroni, as well as a Pulse Sensor Amped. Put together by Jon Witts, it measures live heartbeats and displays the results on a Scroll pHAT HD.

> magpi.cc/heartratemonitor



Make a dashcam

Dashcams can be attached to your car to record video in case of an incident. ZeroView is a mount with suction cups designed to house Raspberry Pi Zero. It includes a mount for your Raspberry Pi Camera Module. The cups enable it to be attached to a car windscreen, or any other type of window.

> magpi.cc/zeroview

“ ZeroView is a mount with suction cups designed to house Raspberry Pi Zero”

Make a mood light

The Mood Light is a Raspberry Pi Zero project kit from Pimoroni. It uses the GPIO pins on Raspberry Pi Zero to connect to a Unicorn pHAT with 32 RGB NeoPixel LEDs. You can set these to a single colour, or fade them through all the colours of the rainbow.

> magpi.cc/moodlight

Smart Calendar

Raspberry Pi Zero 2 W is great for powering an e-ink display, which draws less power than an LCD. The combination is perfect for presenting this always-on calendar that updates once an hour, or at the press of a button. Zonglin Li has infused the front-end with numerous widgets and an LED lights up when the screen is being refreshed.

> magpi.cc/einkcalendar

Raspberry Pi Zero 2 W QuickStart Guide

Setting up and using Raspberry Pi Zero 2 W is pretty straightforward

Congratulations on becoming a Raspberry Pi Zero 2 W owner. We're sure you'll enjoy discovering a whole new world of computing and the chance to handcraft games, control robots, build machines, and share your experiences with other Raspberry Pi fans.

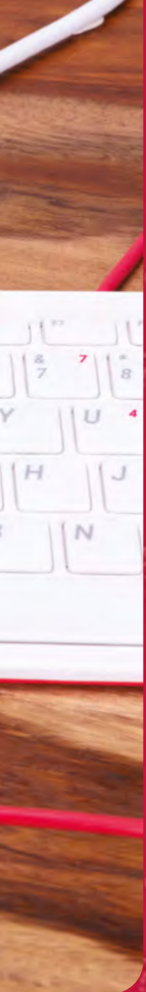
Getting started won't take long: just corral the extra bits and bobs you need on our checklist. To get set up, use Raspberry Pi Imager to set up a card and connect all the cables. This guide will lead you through each step. You'll find Raspberry Pi OS, including coding programs and office software, all available to use. After that, the world of digital making with Raspberry Pi awaits you.

What you need

All the bits and bobs you need to set up a Raspberry Pi computer

8GB microSD card

You'll need a microSD card with a capacity of 8GB or greater. Raspberry Pi Zero 2 W uses it to store the operating system and store programs and files. If you want to reuse an old card, you'll need a card reader: either USB or a microSD to full-sized SD (pictured).



Linux, Mac, or Windows computer

You'll need a Linux PC (such as another Raspberry Pi), Windows Linux PC, or Apple Mac computer, to run Raspberry Pi Imager to download and write Raspberry Pi OS onto a microSD card for Raspberry Pi Zero 2 W.

Power supply

Raspberry Pi Zero 2 W uses the same type of micro USB power connection as many old electronic devices. So you can recycle an old USB to micro USB cable and a smartphone power supply. Raspberry Pi also sells official power supplies (**magpi.cc/products**), which provide a reliable source of power.

USB keyboard

Like any computer, you need the means to type commands, create code and documents, and otherwise control Raspberry Pi. You can use a Bluetooth keyboard, but the initial setup process is much easier with a wired keyboard. Raspberry Pi sells an official Keyboard and Hub. **magpi.cc/keyboard**

Display and HDMI cable

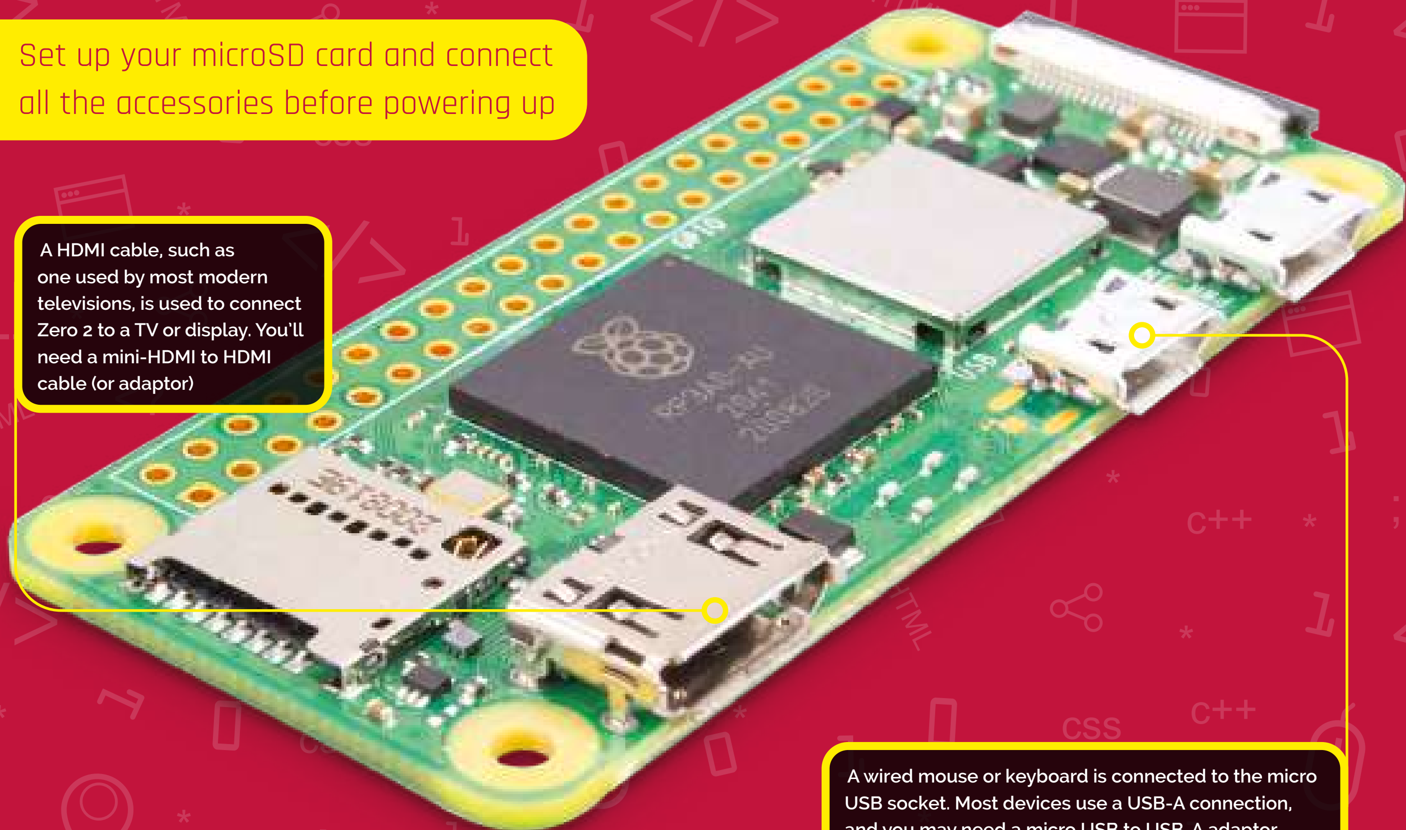
A standard PC monitor is ideal, as the screen will be large enough to read comfortably. It needs to have an HDMI connection, as that's what's fitted on your Raspberry Pi board. Raspberry Pi Zero 2 W needs a mini HDMI to HDMI cable (or adaptor). **magpi.cc/minihdmi**

USB mouse

A tethered mouse that physically attaches to your Raspberry Pi via a USB port is simplest and, unlike a Bluetooth version, is less likely to get lost just when you need it. Like the keyboard, we think it's best to perform the setup with a wired mouse. Raspberry Pi sells an Official Mouse (**magpi.cc/mouse**).

Set up Raspberry Pi Zero 2 W

Set up your microSD card and connect all the accessories before powering up



A HDMI cable, such as one used by most modern televisions, is used to connect Zero 2 to a TV or display. You'll need a mini-HDMI to HDMI cable (or adaptor)

A wired mouse or keyboard is connected to the micro USB socket. Most devices use a USB-A connection, and you may need a micro USB to USB-A adaptor (**magpi.cc/usbadaptor**).

01 Get it connected

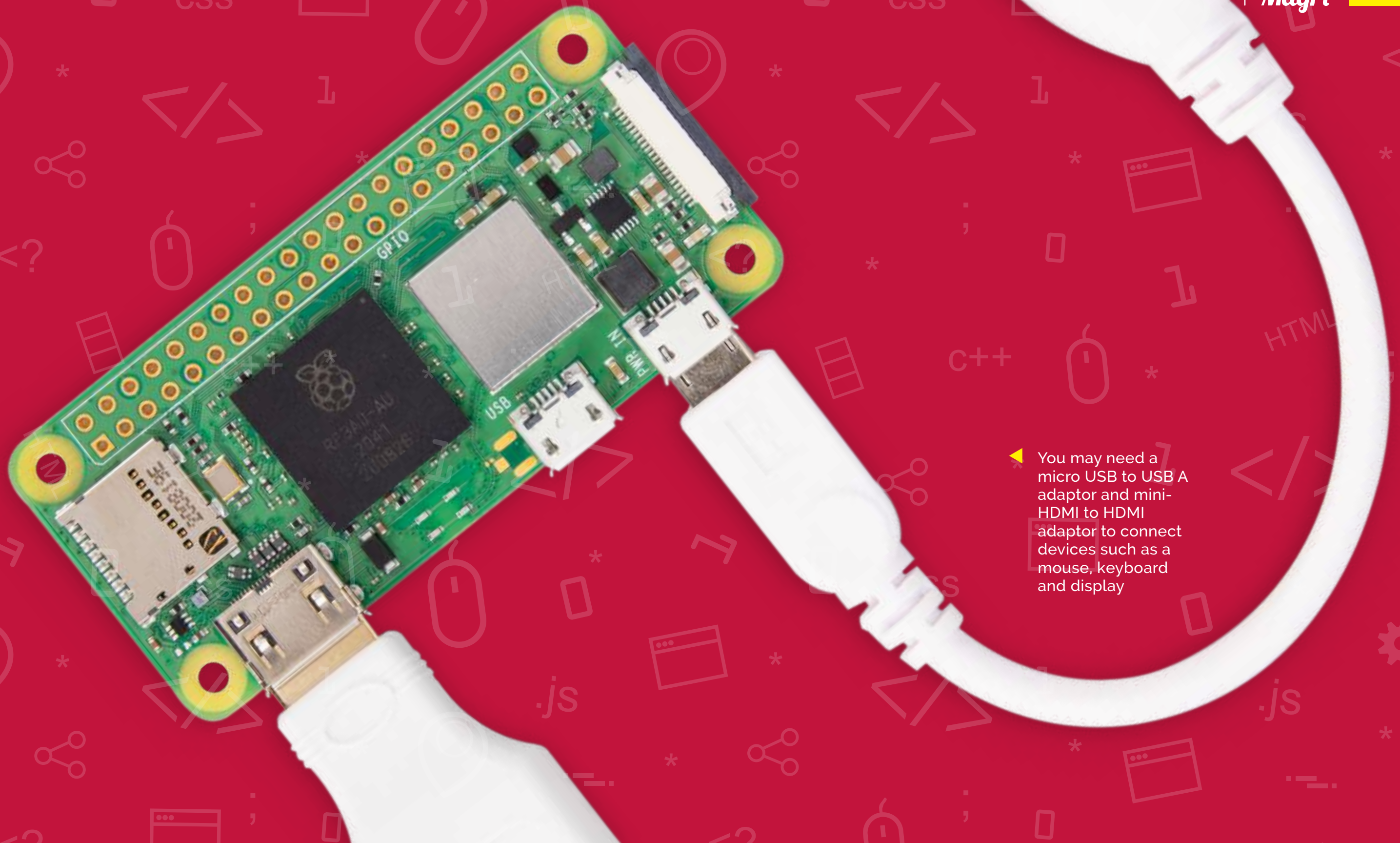
As you're setting up a Raspberry Pi Zero 2 W, you'll need to use a USB-A to micro USB-B cable (or adaptor) to connect a keyboard to the smaller connection on a Raspberry Pi Zero W. One is included with the official keyboard, which also has additional USB-A ports to accept the mouse and other accessories. Or you can use a cable with your own keyboard.

02 Mouse and keyboard

You can either connect your mouse to a USB socket on your keyboard (if one is available), then connect the keyboard to the micro USB socket (via the micro USB-A to USB-B adapter). Connect your mouse to one of these if possible. If not, you'll need a USB HUB so you can connect both the keyboard and mouse to the single USB-B socket.

03 More connections

Now connect your full-sized HDMI cable to the mini-HDMI to HDMI adapter, and plug the adapter into the mini-HDMI port in the middle of



You may need a micro USB to USB A adaptor and mini-HDMI to HDMI adaptor to connect devices such as a mouse, keyboard and display

your Raspberry Pi Zero 2. Connect the other end of the HDMI cable to an HDMI monitor or television.

04 Set up the software

Now you've got all the pieces together, it's time to install an operating system on your Raspberry Pi so you can start using it. Download Raspberry Pi Imager from **magpi.cc/imager**. This utility is available for Windows, macOS, and Linux computers, so choose the relevant version for your system.

05 Write the OS to the microSD card

Attach your microSD card to your PC or Mac computer, and launch Raspberry Pi Imager. Click the 'Choose OS' button to select which operating system you would like to install. The top option is Raspberry Pi OS (32-bit). With an OS selected, click the 'Choose SD card' button and select your microSD card (typically there will be just one option).

Finally, click the 'Write' button and wait while the utility writes the selected OS to your card and then verifies it. When complete, you may remove the microSD card.

> " When Raspberry Pi OS first loads, you will need to set a few preferences "

06 Assemble your Raspberry Pi

Now it's time to physically set up your Raspberry Pi. Plug your PC monitor into the mains. Remove the microSD card from the SD card adaptor and slot it into the underside of your Raspberry Pi Zero 2 W.

07 Power up

Plug in your Raspberry Pi power supply and, after a few seconds, the screen should come on. Raspberry Pi OS will boot up. When Raspberry Pi OS first loads, you will need to set a few preferences. Click Next, when prompted, then select your time zone and preferred language, and create a login password. You're now ready to get online. Choose your WiFi network and type any required password. Once connected, click Next to allow Raspberry Pi OS to check for any OS updates. When it's done so, it may ask to reboot so the updates can be applied. You're all set to start enjoying computing with your very own Raspberry Pi Zero 2 W.

Part 01

Sensory world: Build a fire and gas leak alarm system

Enable Raspberry Pi to detect flames and gas leaks to raise an alarm

MAKER

Phil King

Longtime contributor to *The MagPi*, Phil is a freelance writer and editor with a focus on technology.

@philkingeditor

Humans experience the world through a range of senses. This enables us to be aware of whatever's happening in our environment so that we can react to it.

So, it makes sense (no pun intended) that your Raspberry Pi would also benefit from being able to sense things. Fortunately, this is made possible using a wide variety of electronic sensors. In this series, we'll explore some of the most commonly available sensors and their use cases.

To start off, we'll build a simple fire and gas leak alarm system using a couple of sensors. When either hazard is detected, a visual and audible alert will be triggered.

Figure 1

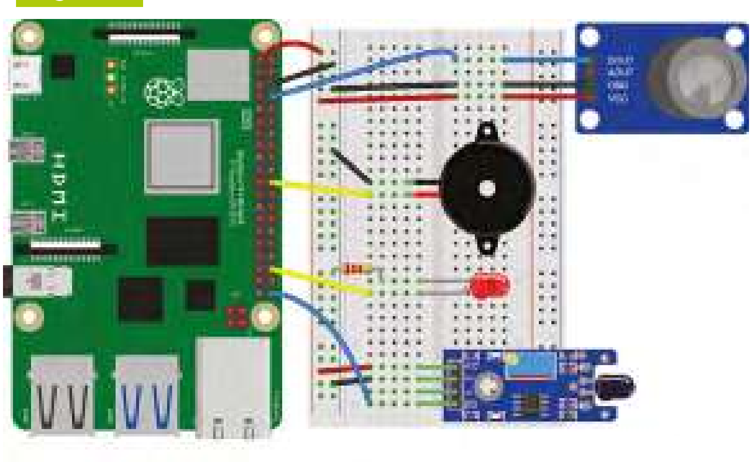


Figure 1 The wiring diagram for the complete alarm system, including two sensors, LED, and buzzer

01 Connect flame sensor

For this tutorial, we're using the flame sensor from the Waveshare Sensors Pack, available in the UK from The Pi Hut (**magpi.cc/wavesensors**), and also sold separately, but any similar sensor should work in a similar way. It uses PIR (passive infrared) to detect a change in temperature nearby. While the sensor has pins for both analogue and digital outputs, we only need the latter for our alarm, as a nearby flame will set the digital output to high.

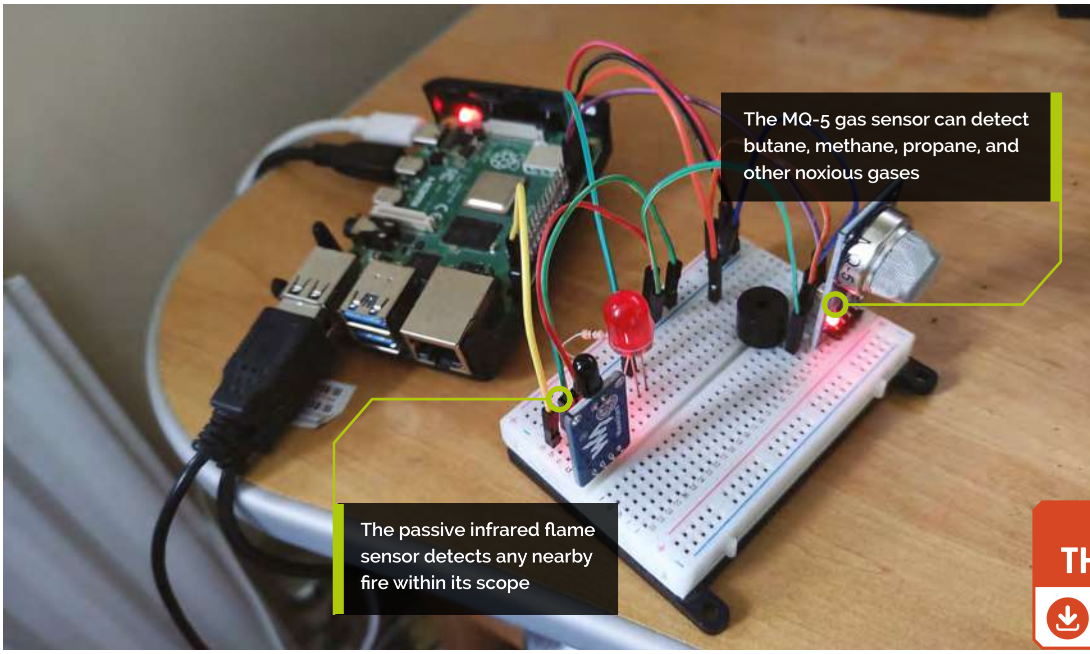
With the power turned off, connect the flame sensor to Raspberry Pi, as in **Figure 1** (where it's at the bottom). We're powering it from Raspberry Pi's 3V3 pin, grounding it with a GND pin (both via the breadboard side rails), and the digital output (marked DOUT on the sensor) is going to GPIO 21.

02 Flame test code

With the flame sensor wired up, turn on Raspberry Pi. You should see the sensor's red power LED light up if it's connected correctly.

To start with, we'll create a simple Python program, as in the **flame_test.py** listing, to read the sensor and print out a message when triggered, to check that it's working correctly. From the desktop menu, go to Programming and open the Thonny IDE to start coding.

To simplify the setup, we're using the GPIO Zero library rather than RPi.GPIO; at the top of



DOWNLOAD THE FULL CODE:

magpi.cc/github

our code, we import the Button method from it. We'll use this to sense when the digital output from the sensor is high, in effect the equivalent of a push-button being pressed. As it's connected to GPIO 21, we assign the `flame` object to this with `flame = Button(21)`.

In an infinite `while True:` loop, we check whether the pin is high (`if flame.value == 1`) and set the message (`msg1` variable) that we'll be printing to the Thonny Shell area accordingly. In our print statement, we add the `end = "\r"` parameter so that the message is always printed on the same line, which is a lot neater than using a new line each time.

03 Flame on!

Now it's time to test our sensor by putting a flame near to it to see if it triggers our alarm message. We used a disposable lighter for this, but you could just light a match. Always be careful with fire, though, and don't get the flame right next to the sensor as it's not fire-proof!

Run your **flame_test.py** Python code and then move the flame towards the sensor. Ours triggered at around 30 cm distance, but the sensitivity can be altered by using a small crosshead screwdriver to adjust the screw on the sensor board – other flame sensors may have a potentiometer rotary knob on board for this purpose.

When the flame is close enough to trigger the sensor, another red LED lights on ours to indicate this. The message printed in the Thonny Shell area should change from 'No fire' to 'Fire!'.

flame_test.py

Language: **Python**

```python
from gpiozero import Button

flame = Button(21)
msg1 = ""

while True:
    if flame.value == 1:
        msg1 = "Fire!  "
    else:
        msg1 = "No fire"
    print(msg1, end = "\r")
```

04 Add a gas sensor

So, we have our flame sensor working correctly. Now it's time to add the gas sensor to our setup. We're using a Waveshare MQ-5 sensor for this, as featured in the Sensors Pack, but other similar gas sensors are available. The tin oxide layer inside the sensor has a high sensitivity to the presence of gases such as butane, methane, and propane.

The sensor has pins for analogue and digital outputs, but we only need the digital output for our alarm. We connect that pin (DOUT) to GPIO14, and the VCC and GNC pins to 3V3 and GND (shared with the flame sensor via the breadboard side rails), as in **Figure 1** (gas sensor at the top).

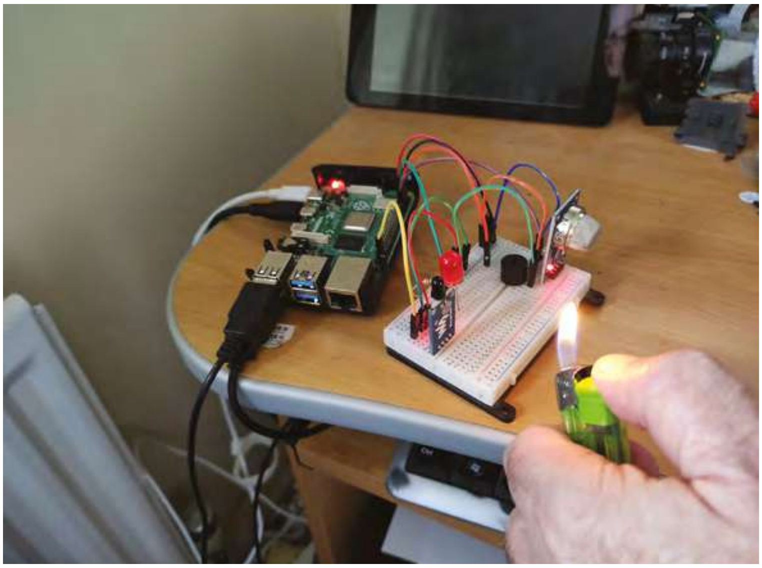
Top Tip

Analogue out

For simplicity, we've used the digital outputs to trigger our alarm. To use the analogue outputs, you'll need to add an ADC chip (e.g. MCP3008) to convert them to digital readings.

You'll Need

- Flame sensor **magpi.cc/flamesensor**
- MQ-5 gas sensor **magpi.cc/gassensor**
- LED
- Active piezo buzzer
- Jumper wires



Testing the flame sensor with a lighter. Be careful not to move it too near the sensor or other components!

Top Tip

Off board

We've placed our sensors on the breadboard, but you could keep them separate from the other components and wire them up from there, with the digital outputs wired directly to the GPIO pins.

flame_gas_test.py

> Language: **Python**

```python
from gpiozero import Button

flame = Button(21)
gas = Button(14)
msg1 = ""
msg2 = ""

while True:
    if flame.value == 1:
        msg1 = "Fire!   "
    else:
        msg1 = "No fire "
    if gas.value == 1:
        msg2 = "Gas leak!"
    else:
        msg2 = "No gas   "
    print(msg1, msg2, end = "\r")
```

05 Add gas to code

While we could create a new program to test the gas sensor separately, we'll add it in to our previous code as it will then be simpler to rework it into our final alarm code later.

In the **flame_gas_test.py** listing, we assign the gas object to GPIO 18 with `gas = Button(14)`. As before, we're using the Button method to detect when the pin is triggered and thus set to high.

We add a new `msg2` variable, which we'll set to the message we want to print in the Shell area. This is done by adding some extra lines to our `while True:` loop. If the gas sensor pin is triggered (`if gas.value == 1`), then the message will be set to 'Gas leak!'; if not, it'll be 'No gas'.

We add `msg2` to our print statement, after `msg1`, again including the parameter `end = "\r"` so the message is always printed on the same line.

06 Turn on the gas

Now it's time to test our gas sensor to check it's wired up and working correctly. As before, we're using a disposable lighter for this, pressing the button to release a little gas without igniting it for the flame. You could use gas from a hob or cylinder, but you only need a tiny amount to trigger the sensor, so be sure to turn it off again after a couple of seconds.

Run your **flame_gas_test.py** Python code and then release a little gas in the vicinity of the sensor. As with the flame sensor, the gas sensor's sensitivity can be altered by using a small crosshead screwdriver to adjust the potentiometer screw on the sensor board.

When the presence of gas is detected and the sensor is trigger, another red LED lights on ours to indicate this. The second message printed in the Thonny Shell area should change from 'No gas' to 'Gas leak!'.

> "The gas sensor's sensitivity can be altered"

07 Let there be light

If your sensors and code are working correctly, and the correct messages are shown when they're triggered, it's time to move on to the next part. Printed messages are all very well, but for a proper alarm you need a visual and/or audible alert.

For the visual side, we'll add a standard red LED to our setup – ours is 5 mm, but you can use any size. As always when using LEDs, a resistor is required to limit the current to make sure it doesn't receive too much and potentially burn out. With the LED placed in the breadboard, with legs in different unconnected rows, we connect a 330 Ω resistor between the negative (shorter) leg and the ground rail of the breadboard. The positive (bent, longer) leg is connected to GPIO 16 on Raspberry Pi (as in the **Figure 1** wiring diagram).

The Waveshare flame sensor has both analogue and digital outputs. Sensitivity can be adjusted with the screw

The MQ-5 gas sensor can detect several noxious gases in the air and is highly sensitive

08 Get a buzz

For our audible alert, we're using a small active piezo buzzer to make a beeping noise. The buzzer will have a longer positive leg and a shorter negative one; their positions may also be marked on top of the buzzer itself. Connect the negative pin to the breadboard's ground rail and the positive pin to GPIO 25 (as in **Figure 1**).

09 Alarm code

With everything wired up as per **Figure 1**, you're now ready to program your fire and gas alarm. In the final code, **fire_gas_alarm.py**, we add LED and Buzzer to the gpiozero imports at the top. We also import sleep from the time library, to use as a delay.

We create two functions, one for each type of alarm: `fire_alarm` and `gas_alarm`. Each uses a `for` loop which toggles the LED and buzzer on and off a set number of times, with a 0.5 sleep delay each time.

Finally, in a `while True:` loop, we check the pin values and trigger the relevant alarm when the pin receives a signal from the sensor. If neither is triggered, we show the default message and ensure the LED and buzzer are turned off.

10 Sound the alarm

Now to test the alarm system. As before, try positioning a flame near the fire sensor to check that the alarm is triggered, in which case the LED will blink and the buzzer will beep. Do the same for the gas sensor by releasing a small amount of gas; the alarm will sound again. Each time, an appropriate message will show in the Shell area.

Taking it further

We now have a simple working fire and gas alarm. To make the alert more obvious, you could add a relay switch to turn on a 12 V tower light with buzzer (e.g. **magpi.cc/towerlight**). You could also send an email or push notification alert to your phone whenever the alarm is triggered.

Next time we'll create an intruder alarm using noise and laser sensors. See you then.

fire_gas_alarm.py

Language: **Python**

```python
from gpiozero import Button, LED, Buzzer
from time import sleep

flame = Button(21)
gas = Button(14)
led = LED(16)
buzzer = Buzzer(25)

def fire_alarm():
    print("Fire!     ", end = "\r")
    for i in range (10):
        led.toggle()
        buzzer.toggle()
        sleep(0.5)

def gas_alarm():
    print("Gas leak!", end = "\r")
    for i in range (10):
        led.toggle()
        buzzer.toggle()
        sleep(0.5)

while True:
    if flame.value == 1:
        fire_alarm()
    elif gas.value == 1:
        gas_alarm()
    else:
        print ("All OK    ", end = "\r")
        led.off()
        buzzer.off()
```

Warning!
Fire and gas

Never play with fire! Be careful when testing your alarm and don't position the lighter flame too close to the sensors or other components. Only release a small amount of gas and do it in a well-ventilated area. Do not rely on this DIY alarm for your safety. In the event of a real fire or gas leak, call the relevant emergency service.

magpi.cc/firesafety
magpi.cc/gassafety

Turn Keybow 2040 into a stream deck

Reprogram the RP2040 auxilliary keyboard into a powerful macro keyboard for streaming and more

MAKER

Rob Zwetsloot

Rob is amazing. He's also the Features Editor of *The MagPi*, a hobbyist maker, cosplayer, comic book writer, and extremely modest.

magpi.cc

You'll Need

- Keybow 2040 **magpi.cc/keybow2040**
- OBS **obsproject.com**
- VoiceMod **voicemod.net**
- USB-C to USB-A cable

Streaming is fun. Whether you're doing art, playing games, or just chatting with followers, it can be great to interact with folks while doing something else.

It is also quite complex to stream. Starting with a complicated OBS setup to stream your game, and ending up with loads of extra tabs and audio mixers thrown in, there can be a lot to control and maintain while you're live, and even more to do if you want to use silly audio effects or zooms, etc. This is where a stream deck comes in, allowing for short cuts for various functions at the press of a button. A lot of dedicated stream deck products can be pretty expensive, though, so let's make something a lot cheaper and far more customisable.

01 Construct Keybow 2040

The Keybow requires some building before you use it – it's fairly simple though. You attach some standoffs, a middle layer, add the switches to the final plate, add the keycaps, and then sandwich it all together. There are some illustrated instructions here: **magpi.cc/keybowbuild**. You should be done before an episode of *The Simpsons* is over.

For stream deck use, we also have a 3D-printed switch stand that it fits onto nicely, keeping the keys in eyesight (which is important, as we'll explain later).

02 Basic Keybow 2040 usage

Plugging your newly built Keybow 2040 into your PC will activate the pre-installed code that turns it into a numpad using a basic HID (human interface device) script. The LEDs in the keys are turned off, but go green as they are pressed.

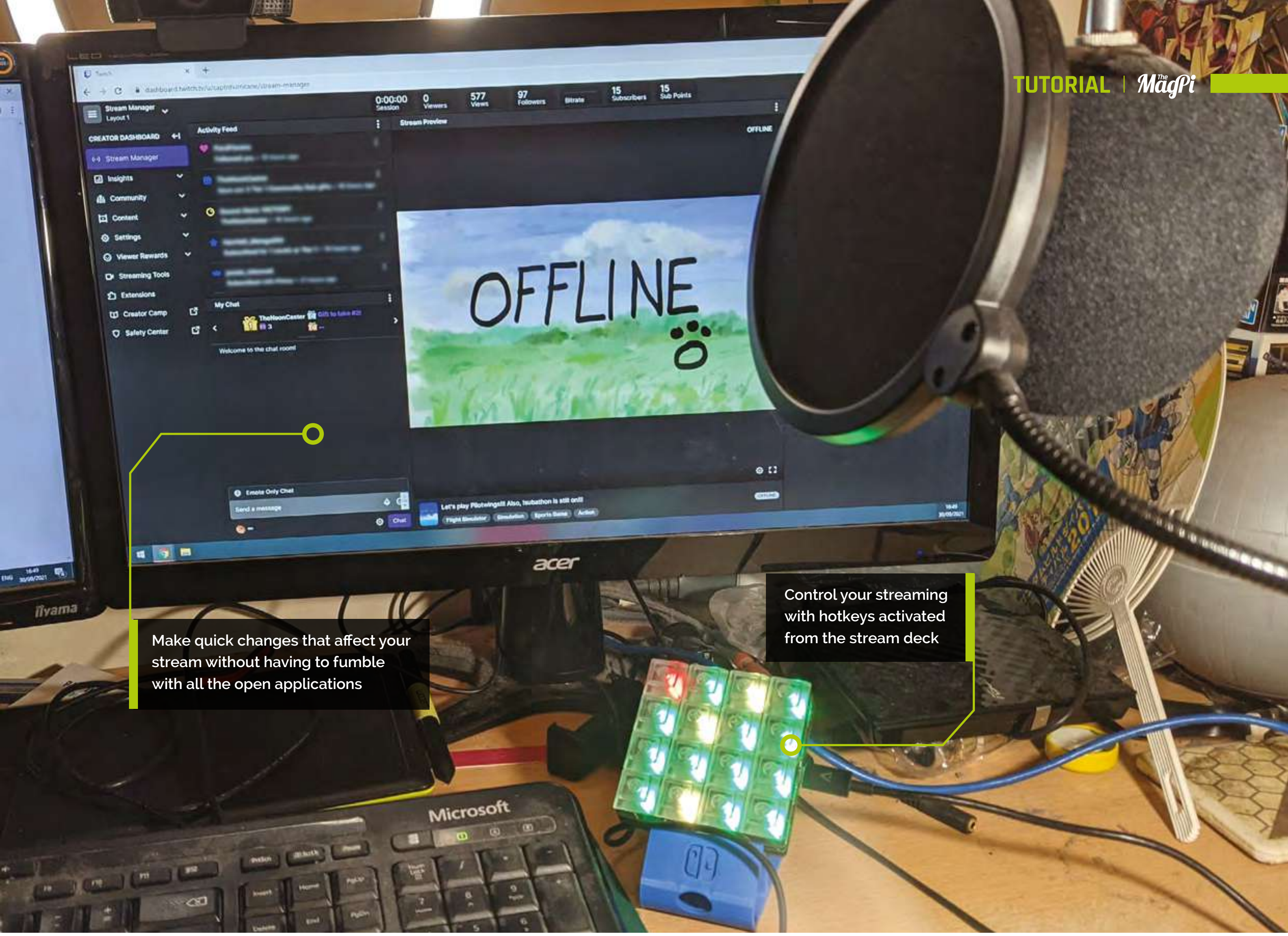
As well as being usable while plugged in, you can also access the storage inside and find different examples in the **examples** folder. To use one, merely copy it to the top Keybow directory and change the file's name to **code.py**.

> "A lot of dedicated stream deck products can be pretty expensive"

03 Customise your code

The **code.py** file is what tells the Keybow 2040 how to operate. Looking through the examples, you'll be able to see how a lot of the individual parts work together. A full list of the functions can be found on the Keybow GitHub here: **magpi.cc/keybowgit**.

As it runs on CircuitPython, based on MicroPython, a lot of Python knowledge and skills are transferable while writing any custom code. With one caveat though – using sleep is not recommended by Pimoroni, and it did not work with this build either. Still, there's a lot you can do.



It's very quick to prototype and test your custom code as well – opening, editing, and then saving the **code.py** file will immediately reset Keybow 2040 so that you can test out your handiwork.

04 Keyboard shortcuts

The basic keys used by the default code should be altered if you plan to use this as a stream deck. They function as a numpad, and while they will work as hotkeys, they may also be used normally by the software you're running. Luckily, the Arduino key library allows for multiple keys to be sent at once:

```
keyboard.send(Keycode.LEFT_CONTROL, Keycode.
KEYPAD_ZERO)
```

In our code, we're still using numpad keys, but we've added the left **CTRL** key to the combo. You could easily use a different key, such as the scroll lock, or multiple keys. Just check to make sure the key combos you're planning on using aren't being used in other programs – we don't want any **ALT+F4** incidents while streaming games.

05 LED colours

We've opted for all the keys to be lit up while the stream deck is connected, and this can be done using a simple `keybow.set_all()` command, where an RGB code is placed in the brackets.

For our code we're using green (0, 255, 0), yellow (255, 255, 0), and red (255, 0, 0), and you can easily add or modify the colours as you see fit. All the keys light up green when the Keybow is on (mostly to remind us that it is), and keys will turn yellow once activated. The red is used for the mute and voice change button when activated, just to differentiate them.

For other colours, blue is a good one (0, 0, 255), as well as white (255, 255, 255). If you need another shade, simply enter the name of it followed by 'rgb' into your search engine of choice.

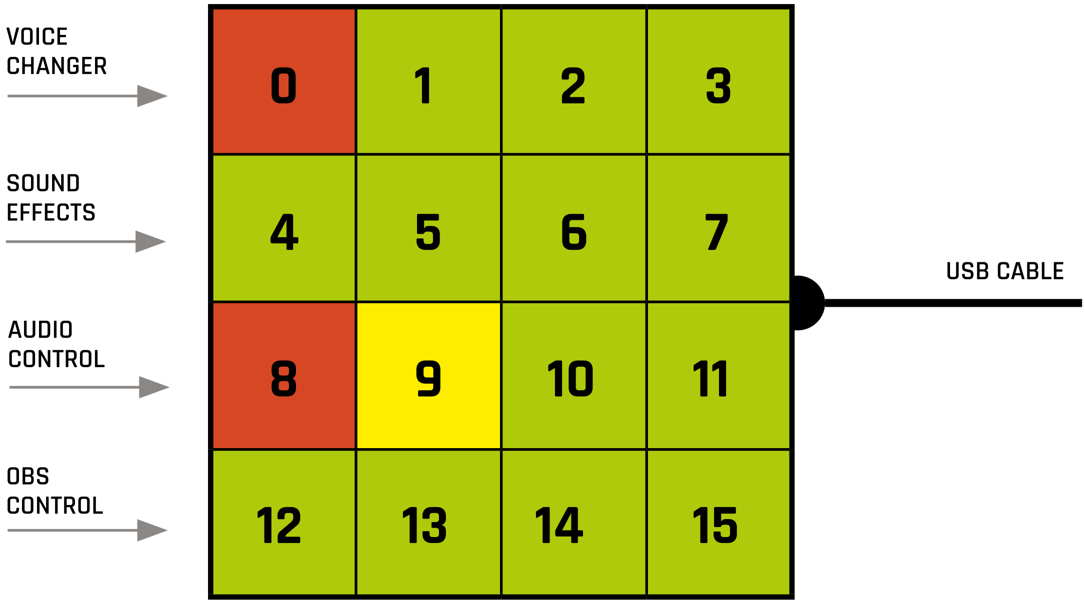
06 When pressed

One of two most important parts of our code is the `press_handler` function. As the code cycles, it waits for a key to be depressed before it starts doing anything. It gets told which key is being depressed, finds out which part of the key

Top Tip

Manual combos

The key combos you've set up can also still be done on your regular keyboard, in case you need a backup way to use a hotkey.



Here's how the keys are laid out on Keybow 2040 related to the code

Top Tip

OBS focus

You may find some difficulties using hotkeys with OBS – make sure the window is in focus for them.

combo it should use, and then sends that as a key press.

We also have an `if` statement so that a specific key turns to yellow when depressed, as all this button does is stop any sound effects from playing – it can easily be applied to other keys by modifying this `if` statement.

07 When released

By having separate press and release functions, we save the code treating a button-press as multiple events. We can also define what happens when the button is released; in this case we change the colour of the key to indicate that it's been used, as well as updating the `keytoggle` list, which we use to remember the state of the key (on or off, 1 or 0)

As we have many different buttons doing different things, we've racked up a long `if` statement – most are simply 'change between two colours'; however, one of them treats a series of three keys differently: `elif key.number >= 1 and key.number <= 3`. Only one may be lit yellow at a time, and we'll be using this to select a type of voice modification.

The `and` here also works as a Boolean operator, where AND means only if it satisfies both requirements, and OR means if it satisfies at least one requirement.

08 Planning your layout

For the code in this tutorial, we've set up the keys in categories, as shown in **Figure 1**. Keys 0–3 on the top line handle voice changing, the second row we use for sound effects, the third line for muting and stopping sound effects, and the last line for OBS hotkeys.

Remember, the keys go from 0 to 15 only when the Keybow is orientated with the USB cable sticking out of the right side.

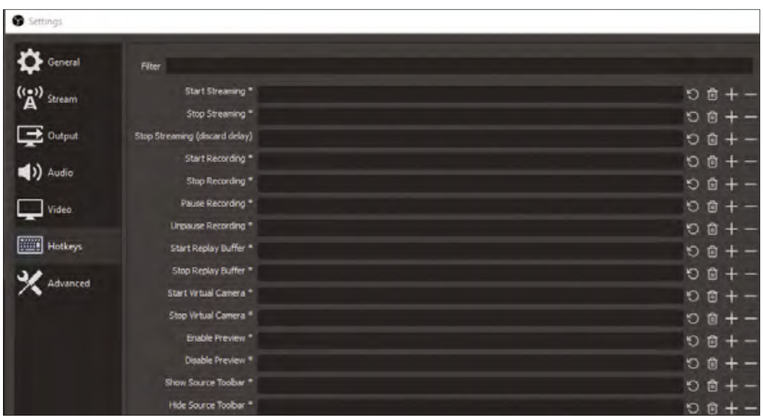
Depending on your stream deck uses, you can make your own layout; however, be aware of the placement when programming.

09 Special keys

Our mute key and voice change key both turn red when activated. These are both important keys in the setup and if you're a bit forgetful, like us, you may not realise that the voice change key is still on until a helpful member in chat points out that you sound like a robot.

They broadly function the same as any other key pressed; however, the differing colours do help

Adding hotkeys to OBS is very easy, and it can control just about every aspect



DOWNLOAD THE FULL CODE:

magpi.cc/streamdeck

◀ The free version of VoiceMod offers many features, and they all work with hotkeys

when you're trying to quickly bump a key while drifting a Ferrari around LA.

10 Toggle keys

The keys 1, 2, 3 we are treating as toggle keys, as explained above. Here is how the code works:

```
key.set_led(*rgb2)
keytoggle[key.number] = 1
```

This part sets the key to yellow, and changes the `keytoggle` list to on/1 for that key.

```
if key.number != 1:
    keys[1].set_led(*rgb1)
    keytoggle[1] = 0
```

This part checks to see if key 1 was not pressed, and the following two statements check for keys 2 and 3. If those keys were not pressed, the colour is returned to green, and the `keytoggle` is set to off/0.

> "Our mute key and voice change key both turn red when activated"

11 OBS hotkeys

Just about every element in your OBS setup will be able to have a hotkey attached – whether it's simply changing to a scene, showing and hiding an image, or to stop streaming. From OBS, click Settings and then select the Hotkeys tab.

Click on the bar next to the function you'd like to hotkey, and then press the desired Keybow key. It will not only set the hotkey but also tell you what it is – a good way to check that your code worked properly.

12 VoiceMod hotkeys

VoiceMod is a great piece of software for doing fun voice changers, playing sound effects, and generally controlling the entire audio input for your stream. Selecting a modifier, you'll see a button to add a hotkey - click that and use the desired Keyboy 2040 key. The same is true for any sound effects or clips you upload.

In the Settings, you can also add other hotkeys – for the code we've used, key 0 is for voice changer toggle, key 8 for mute/unmute, and key 9 for stopping any sounds.

▼ Elevating and angling a stream deck can make it easier to press buttons… and remember it's there

Part 01

Connect retro MIDI hardware to Raspberry Pi

External MIDI sound cards like the Roland MT-32 were the peak of computer music in the 1980s and 1990s. They still work just as well today

MAKER

K.G. Orphanides

K.G. holds that digital audio hardware peaked in the 1990s and is prepared to die on this preposterous hill.

@KGOrphanides

External MIDI sound cards were the pinnacle of 1980s and early 1990s computer music production and playback. We've used a Roland MT-32 and SC-55 in this tutorial, but you can use most external MPU, MT-32, GM, GS, or XG MIDI devices from the era. Although some current hardware MIDI sequencers are marketed to musicians, modern consumer audio interfaces usually lack integrated MIDI voices, and even early 2000s implementations can suffer variable General MIDI sound bank quality. However, the Serda 3XMA (**magpi.cc/x3m**) is a well-regarded and, at €79 (£68), relatively inexpensive modern USB MIDI interface with sound banks designed for backwards compatibility with the SC-55.

01 Hook me up

If you're using original 1980s or 1990s hardware, start by connecting your MIDI-to-USB cable. This has a standard USB Type-A plug on one end, and a pair of 5-pin DIN MIDI plugs on the other.

Above Here, we're using Jack's Connect interface to connect Qtractor to our MIDI interface, and our Vortex Wireless MIDI keytar to Qtractor. Think of the lines as patch cables

Connect the MIDI OUT plug on the lead to the MIDI IN port on your audio interface. The MIDI IN plug on the cable goes into the MIDI OUT port on the sound card. Power up your external audio device and make sure you've connected headphones or speakers – audio connections vary. Our MT-32 rev 0 has only a pair of 1.4-inch TRS outputs, so make sure you have appropriate cables and adapters. On some devices, you should keep the MIDI device volume low to avoid crackling. Similarly, if you accidentally attach the MT-32's right output to your speaker's left channel and vice versa, your sound will be overwhelmed by static upon playing anything.

02 Power me on

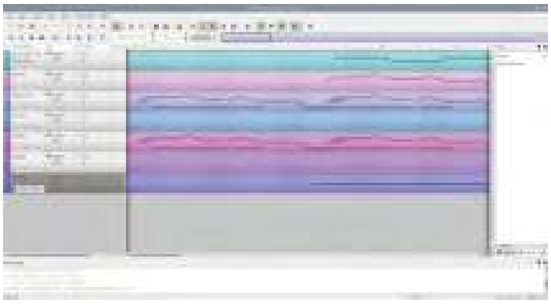
Connect the USB-to-MIDI cable to a USB port on Raspberry Pi and power on. Raspberry Pi OS should detect it as an ALSA device. We'll check that it's there and then run a couple of tests to make sure it's plugged in, discover its port number, and play a MIDI file. Open a Terminal and enter:

```
amidi -l
sudo apt install pmidi
wget wiki.ultimacodex.com/images/8/80/
Stones.mid
pmidi -l
pmidi -p 20:0 Stones.mid
```

The port number after `-p` should be whatever `pmidi -l` reports your device as. If pmidi doesn't list any devices, reboot Raspberry Pi. We're going to need MIDI sequencer software next, so let's grab Qtractor and Jack right away. At the Terminal, enter: `sudo apt install qjackctl qtractor`.

You'll Need

- USB to 5-pin MIDI cable
- An external MIDI sound card
- ScummVM **magpi.cc/scummvm**
- DOSBox-X **magpi.cc/dosboxx**



Above Qtractor is a simple but effective Digital Audio Workstation (DAW) capable of handling MIDI files and composition

03 Jack in

To minimise latency, we're going to be using the Jack Audio Connection Kit, which should have been fully installed as a dependency of qjackctl. At the Terminal, type:

```
qjackctl
```

Now click Setup and, at the right of the screen, set MIDI Driver to seq from the pull-down. Leave the Advanced tab at default settings – if you've previously edited your Jack profile, you may need to reset the Output and Input Device fields to '(default)'. Do not select your USB MIDI interface here, as Jack is looking for PCM audio support and will crash without it.

From Raspberry Pi's applications menu, go to Sound and start Qtractor.

04 Import to DAW

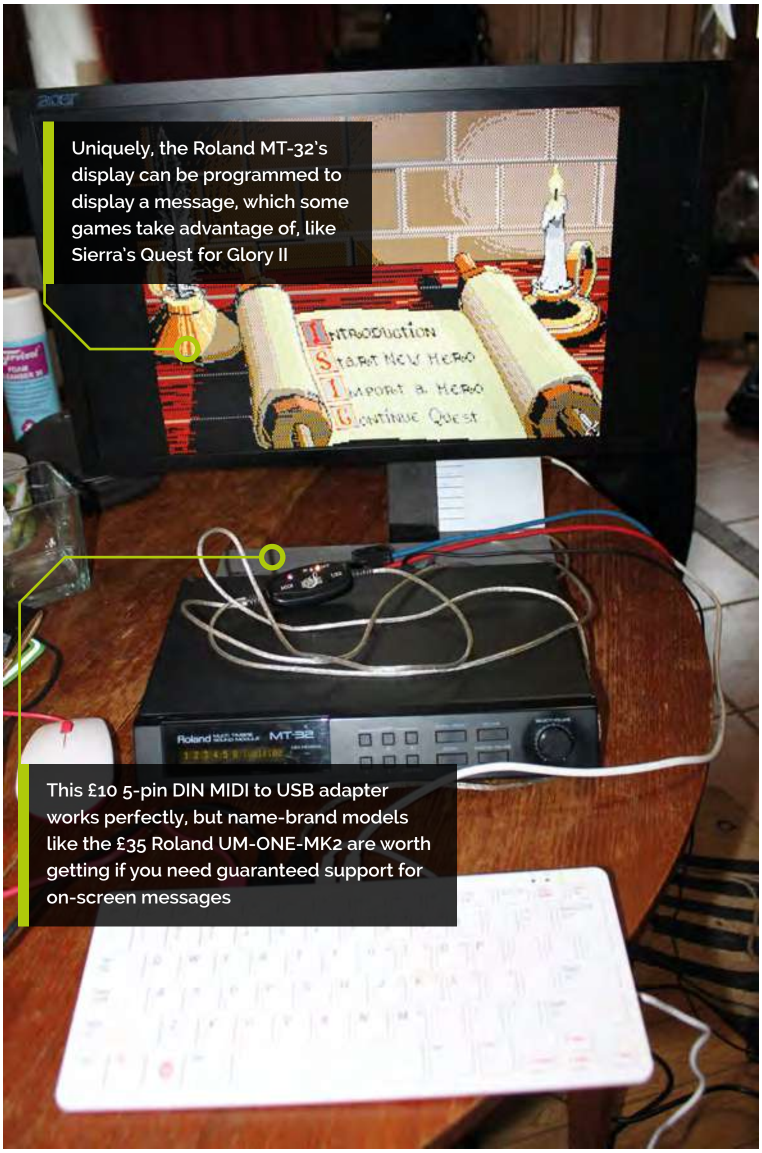
Time to try our Digital Audio Workstation. In Jack's main window, select Connect and click the ALSA tab. Select Qtractor from the left column (Output ports) and USB MIDI Interface from the right (Input ports), then click Connect. A line should appear between them.

In Qtractor, in a new project, go to the Track menu > Import tracks > MIDI. Select the **Stones.mid** file we were using earlier. This should open several tracks in Qtractor. Click the <| button to make sure the play head is at the beginning, then hit play. You should be able to hear and see our MIDI file.

05 How about MIDI input?

You can manually use Qtractor's piano roll to create and edit music. But if you have a MIDI instrument with either a USB connection or another 5-pin DIN and USB adapter, you can use that as a control surface to record or play music directly through your retro MIDI device. Plug in your instrument, create a new Qtractor file, and go back to the ALSA tab in Jack's Connect window – restart Jack if your MIDI instrument doesn't show up.

Connect your MIDI instrument on the left to Qtractor on the right, and Qtractor on the left is connected to your USB MIDI Interface on the right.



Uniquely, the Roland MT-32's display can be programmed to display a message, which some games take advantage of, like Sierra's Quest for Glory II

This £10 5-pin DIN MIDI to USB adapter works perfectly, but name-brand models like the £35 Roland UM-ONE-MK2 are worth getting if you need guaranteed support for on-screen messages

06 Your own composition

Go back to Qtractor, right-click in the left-hand pane, and select Add Track. Select Type: MIDI, Channel: 2, and Program: 17 for an MT-32 harpsichord – or Program: 7 for GM hardware. Click OK and play. If that works, make sure you're at the start of the track, press record, play, and you can now record real-time MIDI input. Give each track its own channel – the MT-32 supports up to

Above The MT-32 defaults to receiving MIDI on channels 2–9; press the MASTER VOLUME and 5 buttons at the same time, then tap 1 to switch it to a more conventional channel 1–8 configuration

eight simultaneous channels, plus percussion. The SC-55 and other General MIDI cards support up to 16. You'll notice that Qtractor doesn't display a patch map with instrument names. You can find an MT-32 list at **magpi.cc/mt32** and a list of the GM GS instruments used by the later devices on Wikipedia (**magpi.cc/midievents**).

07 Add real MIDI to ScummVM

ScummVM allows you to play a wide range of retro and even modern games, from the classic LucasArts titles it was first designed for, to adventures by Sierra, RPGs by SSI, and modern AGS adventures.

Go to **magpi.cc/scummvmdownloads** and download the 'Raspberry Pi 3/4/400 with Raspberry Pi OS 32 bits (armhf)' AppImage. Then download Beneath a Steel Sky – Freeware Floppy Version from **magpi.cc/scummvmgames** and extract it somewhere convenient.

In your file manager, right-click on the ScummVM AppImage, select Properties, click the Permissions tab, and set the Execute pull-down to Anyone. Click OK, then double-click the file to launch it.

08 MIDI configuration for ScummVM

ScummVM uses ALSA to output to your MIDI device. Although they can work together, stop Jack to avoid potential conflicts during configuration. In ScummVM, click Options. In the Audio tab, select your USB MIDI Interface as your preferred output device. In the MIDI tab, select it as your GM device. Finally, in the MT-32 menu, select your MIDI interface as the device and, depending on what hardware you have, tick either True Roland MT-32 or, if you're using an SC-55 or similar GM/GS

Above More readily available than the MT-32, the Roland SC-55 and its successors use a particularly lovely-sounding General MIDI sound bank

synth, Roland GS device. Click OK to return to the main screen.

Select Add Game, navigate into the directory where you extracted Beneath a Steel Sky, and click Choose. Select the game's entry and click Start. The theme music should play through your MT-32, while SoundBlaster-compatible sound effects will come through Raspberry Pi's PCM output.

09 Do it in DOSBox-X

ScummVM can't play everything from that era, so we'll configure DOSBox-X to use our MIDI device, too. Open a Terminal and type:

```
sudo apt install automake libncurses-dev nasm
libsdl-net1.2-dev libpcap-dev libfluidsynth-
dev ffmpeg libavdevice58 libavformat-*
libswscale-* libavcodec-*
git clone https://github.com/joncampbell123/
dosbox-x.git
cd dosbox-x
./build
sudo make install
dosbox-x
```

Top Tip

SC-55 in software

If you want GM sounds without hardware tinkering, download FluidSynth and direct your MIDI output through that via Jack.

MIDI jargon buster	
MIDI	Musical Instrument Digital Interface, an electronic music communication standard
SysEx	System Excclusive messages, used to send instrument patches, messages, and control signals to MIDI devices
MPU	Roland's MIDI Processing Unit range. The most famous was 1984's MPU-401
MT-32	This famous Roland MIDI synth became a byword for PC MIDI audio before the GM standard
GM	General MIDI, a 1991 standardised spec for MIDI-controlled sound generators
GS	General Standard/General Sound, a Roland standard that expands GM with extra sounds
XG	Yamaha's 1994 EXtended General MIDI standard, with up to 480 instruments

At DOSBox-X's Z: prompt, type `exit`. This will create the DOSBox-X config directory, Now we can export a config file for easier customisation. Restart DOSBox-X and, at the Z: prompt, type:

```
CONFIG.COM -all -wcd
exit
```

10 Configure your MIDI settings.

Restart DOSBox-X again. Click on the Main menu at the top left of DOSBox-X's menu bar and select Configuration Tool. Next to mididevice, type **alsa**. For midiconfig, type the device identifier pmidi gave us earlier – **20:0** in our case. If you're using a rev 0 Roland MT-32, you may need to add **20:0 delaysysex** to prevent buffer overflow issues. Save this to the config file we set up earlier and we're ready to try out a game.

11 Obtain and extract Martian Dreams

Go to **magpi.cc/ultima2** and grab the free version of Martian Dreams – you'll need to create a GOG account if you don't already have one. The supplied file requires innoextract to use on Raspberry Pi, so open a Terminal and type:

```
sudo apt install build-essential cmake
libboost-all-dev liblzma-dev
wget https://constexpr.org/innoextract/
files/innoextract-1.9.tar.gz
tar xzf innoextract-1.9.tar.gz
cd innoextract-1.9
mkdir -p build && cd build
cmake ..
make
sudo make install
```

Now `cd` to the directory containing the Martian Dreams installer and type:

```
innoextract 'setup_ultima_worlds_of_
adventure_2_-_martian_dreams_1.0_cs_(28044).
exe'
```

Copy the **MARTIAN** folder to a new **/dos/games** directory in your home. Go to that directory and type:

```
dosbox-x INSTALL.EXE
```

Press **ENTER** to progress through the options until you get to audio settings. Select Roland MT-32 or LAPC-1, then select Yes to save and exit. Exit DOSBox and, back at the command line, type:

```
dosbox-x MARTIAN.EXE
```

12 Tell me more

Although a number of external MIDI sound cards had LCD screens, most just display instrument, volume, and similar data. The exception is the MT-32, which could display any text sent as a SysEx (MIDI system exclusive) message to its tiny screen. A number of games, including titles by Sierra and LucasArts, used this to display references to the on-screen action. For more MT-32 utilities and documentation, check out **magpi.cc/mt32resource**. Roland also continues to distribute software and documentation for its older audio hardware at **magpi.cc/rolandtools**.

See **midi.org/specifications** for more specs and standards

Top Tip

Make your own MT-32

If you can't get an external MIDI device, turn a spare Raspberry Pi into one with the MT32-pi operating system: **magpi.cc/mt32pigit**.

Part 04

Ultimate home server: enhance your network

Make your ultimate server a network genie with these apps

MAKER

PJ Evans

PJ is a writer, software developer and tinkerer. His server has just told him it's time for lunch. This might be getting out of hand.

twitter.com/mrpjevans

In the previous parts of this tutorial, we've taken a Raspberry Pi 4 and turned it into a file sharing, media streaming powerhouse. Next we're taking on your network. There are so many ways to enhance your home (or office) network that we can't hope to fit them into a few pages. However, these steps will point you in the right direction and hopefully inspire you to make your home network smarter, faster, and more convenient to use. Follow along and we'll automate your home, allow you to access your data securely from anywhere in the world, and automatically protect you from nefarious websites.

01 Settle on an address

A key aspect for any server is that you should know where it is. It seems obvious, but a typical home network will assign IP addresses dynamically (a system known as DHCP) and there's a fair chance they will change from time to time. We normally get around this by using zero-config domain names like '.local', but we're going to need a set IP address for our server going forward. The easiest way to do this is to ask your router to reserve an IP address for your ultimate server. We recommend you look at doing this before proceeding. Sorry for the vagueness, but every router does this differently.

02 Home control the easy way

Want to automate your home? Fancy being able to control lights or check a home webcam from anywhere in the world? How about if that software cost you absolutely nothing? Home Assistant is an incredible package that integrates a wide range of inexpensive smart devices such as Philips Hue, Ikea Trädfri, and many more. Best of all, it's open enough that you can create your own activities and integrate your own projects relatively easily. Installation on an existing server requires a few steps, but a great guide has been written by the HA team – **magpi.cc/hacore** – that we have summarised here: **magpi.cc/hasummary**.

03 Getting started with Home Assistant

The first time you run Home Assistant, it'll take a few minutes or so to get its act together as it installs everything it needs. HA will also scan your network for any existing devices you have. Once running, you can access the dashboard at **http://ultimate.local:8123** (change the name if you didn't call your server 'ultimate'). You may be surprised to see what is already detected and available. If not, there are hundreds of plug-ins to extend HA's capabilities. Home Assistant is a massive topic, so you may want to have a read of our tutorial in *The MagPi* #99 (**magpi.cc/99**).

▼ Home Assistant can control hundreds of IoT devices with customisable dashboards like this

Our ultimate server now hosts a whole suite of network features in a tiny case

The Home Assistant app allows you to monitor and control all kinds of IoT devices

04 Add an MQTT broker

We will absolutely forgive you if this title sounds like gibberish. MQTT is a very simple protocol for sending messages from device to device. It is loved by the home automation community, so it's no surprise it's well supported by Home Assistant. The principle is simple: any device can publish its data (e.g. a temperature reading) to an MQTT 'topic' (a label to describe the data stream). Then, any other device can 'subscribe' to the MQTT topic and receive updates in real-time. It makes brewing your own sensors or switches so much easier as you don't have to know about the target.

> "A key aspect for any server is that you should know where it is"

05 A nice kind of mosquito

One of the most popular MQTT brokers (a server that receives and broadcasts MQTT events) is Mosquitto. Installation is very straightforward:

```
sudo apt install mosquitto mosquitto-clients
```

This will install Mosquitto and make sure it's running as a service at all times. To test it's working, open two Terminal windows and run these commands, one in each:

```
mosquitto_sub -t "test"
mosquitto_pub -m "message from mosquitto_pub client" -t "test"
```

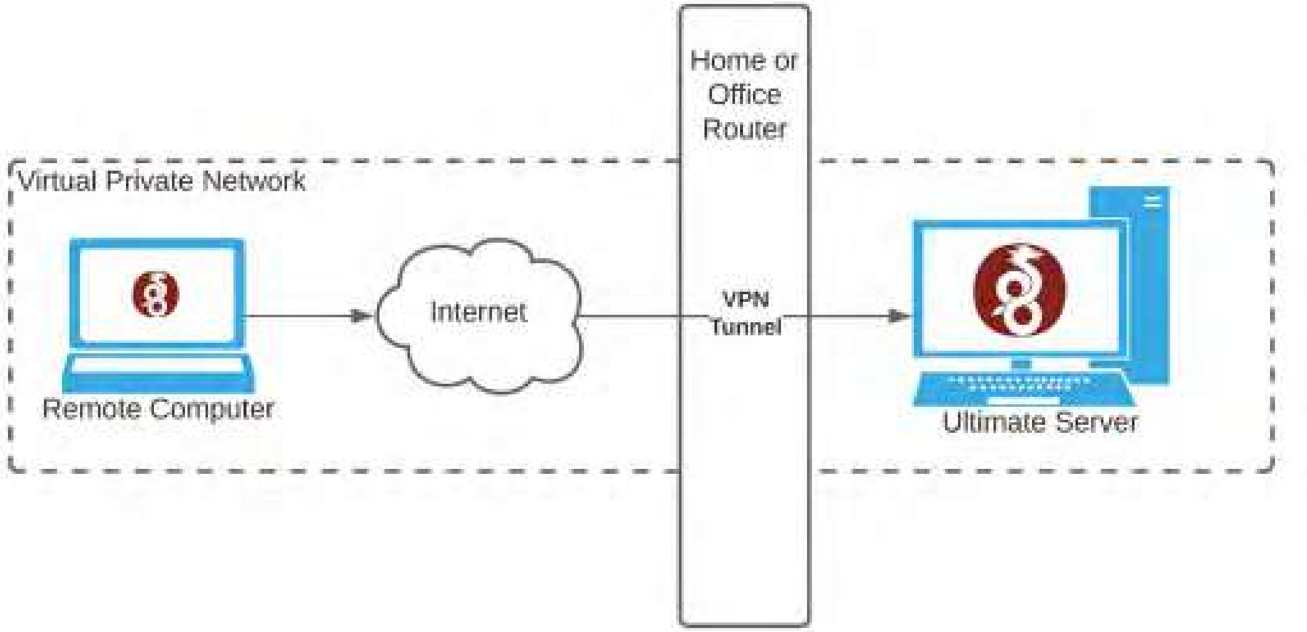
If all is working, you will see a window 'talking' to the other. Now anything that generates MQTT messages can connect with you and your Home Assistant server. Just add the MQTT integration in HA and you can create graphs, alerts, and more.

06 Adding remote access

Previously, we've configured our server to stream audio and video and now we can switch lights, check the living room, feed the cat, and more. So, if you're not home, how do you do this? Remote access to your home network is not something to take lightly, as if you can get access, what stops someone else? The answer is PiVPN, a script that installs a VPN (virtual private

You'll Need

- Raspberry Pi Ultimate Server as built in *The MagPi* #108, #109, & #110



A virtual private network such as Wireguard creates a secure encrypted link to your home devices from anywhere

Top Tip

Building your Home Assistant setup

The Home Assistant community maintains a list of compatible smart devices. These either work 'out of the box' or with a free plug-in: **magpi.cc/haplugin**

network) that you can connect to securely from anywhere in the world. Previously PiVPN only supported OpenVPN, which can be difficult to work with, but now the much simpler Wireguard protocol is supported.

07 Installing Wireguard

For lots of information on PiVPN and Wireguard, head over to The PiVPN Project at **pivpn.io**. To get started with installation, run the following command from a Terminal:

```
curl -L https://install.pivpn.io | bash
```

The installer will take you through a series of questions. An important point is to ensure you have a static IP address as your router will need to know where to find your Wireguard server. When asked what type of server to install, answer 'Wireguard'. Continue through the installer, accepting the default answers. If you don't have a static IP address for your router, you may need to consider a dynamic DNS service such as DynDNS (**account.dyn.com**) so you can easily reach your Wireguard service.

08 Using Wireguard

Before you can use Wireguard to get secure remote access to your network, you need to allow the internet to 'see' it. This is done by creating a port mapping. Your router needs to forward all traffic to port 51820 (default) over UDP to your server. How to do this varies from router to router, so search for your particular router's instructions.

To create users that can access the network, run this command:

```
pivpn add
```

Give a username when prompted and a config file will be created. Copy that config file to the device you want to use to access the network and import it using the free Wireguard client software. Now you can access your home network from a laptop or smartphone from anywhere in the world.

09 Add a web server

The fun thing about web servers is they are actually very simple things. You can place a text file in the home directory and the server will happily serve it anywhere in the world. If you would like to be able to access files through a web browser, you can combine Wireguard and a web server to make access both easy and secure. As an example, we're going to configure access to the media library we have previously created. First, install a web server:

```
sudo apt install apache2
```

After that single line, you would be able to access a simple welcome page at **http://ultimate.local**.

10 Adding directories to your web server

What a web server can provide you with is only limited by imagination. Let's get simple access to our music and videos, which are stored in **/mnt/huge/Media**. Apache has a complex configuration system, but we're going to play a little UNIX trick to make things easier: a symbolic link. This allows us to map one part of the file system to another, creating a kind of short cut.

```
sudo ln -s /mnt/huge/Media /var/www/html/
media
```

Now you should be able to go to **http://ultimate.local/media/** and browse your files. It isn't pretty, but it's a start! Now you can connect to your network securely and access files easily.

> "Pi-hole can protect your entire network, even your smartphone"

11 Avoid the dodgy advertising

One of the most popular Raspberry Pi applications is Pi-hole (**pi-hole.net**), a clever system designed to 'sink' internet advertisements, leaving your browser clear of noise. While we strongly recommend supporting your favourite sites by allowing adverts, many sites contain ads that are not only scams, but can lead to tricking you into downloading malware that can compromise your data. Pi-hole can protect your entire network, even your smartphone. To install it, simply run this command from a Terminal:

```
curl -sSL https://install.pi-hole.net | bash
```

The installer will ask several questions, but is well-designed and you're probably going to be OK to answer every one with the default answer.

12 Starting off with Pi-hole

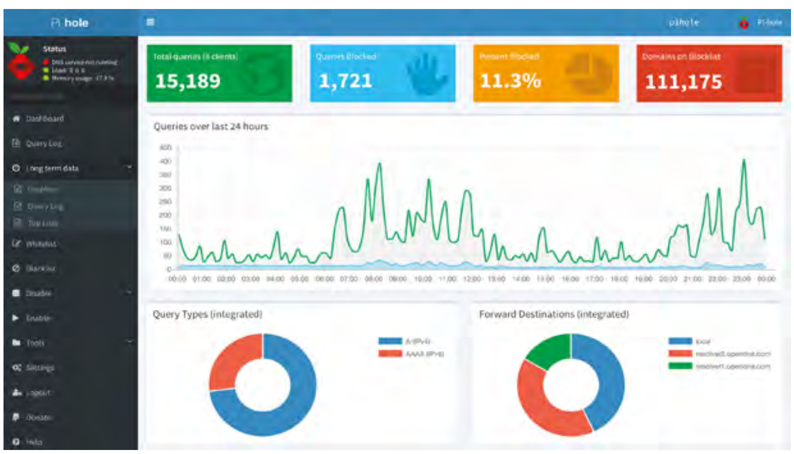
Pi-hole works by looking at DNS requests. These are requests to servers that map domain names such as **raspberrypi.com** to actual servers. It has a maintained blacklist of DNS servers that are for advertising and intercepts any such requests. To do this, Pi-hole must be your DNS server. Here things get complicated. For best results, your new Pi-hole server should also be your network DHCP server (the server that allocates IP addresses to your local devices). This requires some additional setup and we recommend you have a read of *The MagPi* #104 (**magpi.cc/104**).

13 Careful with those ports

A final warning. In this tutorial we've talked about opening ports to the internet using your router's port forwarding features. You may be tempted to just directly open ports to the various services we've installed so you can get access wherever you are. One tip: don't. The reason that services such as Wireguard exist is to ensure that bad actors can't find any easy way to access your stuff. Wireguard uses state-of-the-art encryption. So long as your config file is not compromised, you're more than likely completely safe. If you open your web server to the planet, we guarantee you will have bots sniffing around trying to find access within minutes. VPN isn't easy, but it's worth the time.

14 Next time

Well, we've built a pretty impressive server. File sharing, media streaming, remote access, network protection, and home automation. Next month we will look at how to protect all this hard work, and your precious data, from unexpected loss.



Pi-hole is a network-wide ad blocker that's effective and easy to use

How to write your own emulator in Python

Download the code from GitHub: **wfmag.cc/ wfmag54**

Ever wanted to create your own computer? Here's how to do just that – and even get it to run some classic games

AUTHOR
EDWIN JONES

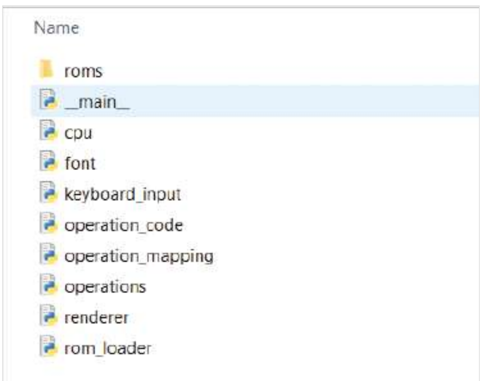
A software engineer at video game developer Mediatonic, Edwin Jones has been working in the games industry for around 16 years. Find out more about him here: **edwinjones.me.uk**

How can we make our own computer with Python? Don't computers need lots of physical things like circuit boards and electronics? Normally yes, they do, but we're going to create something known as a 'virtual machine' – a computer defined completely by software, not hardware. You've probably used virtual machines many times without realising it. Many modern games consoles run older, classic games via virtual machines, also known as emulators.

There are many kinds of emulator, but one of the most fun to start with is the CHIP-8. The CHIP-8 was never a real computer, but that's what makes it fascinating: long before most other games systems, it existed as a program for 8-bit computers in the 1970s. This means there's no hardware to attempt to emulate and quite a simple specification to implement. The CHIP-8 only has 35 opcodes, which is low compared to other computers. You still have all the usual problems to solve – loading binary files, parsing opcodes, and then running them on virtual hardware – but not the other baggage that comes with trying to write emulators for other systems. You also get to finish the project by playing some classic games – what's not to like?

We'll be using Python 3 and Pygame for the project because CHIP-8 isn't very demanding and can be easily emulated. This means it's a good fit for Python, as we don't need to worry so much about performance and can use a language that makes solving the problems at hand a bit easier – we don't need to focus as much on smaller details.

A simple way to build such a device in code is with a class representing the CPU, so let's go ahead and create a `Cpu` class. Create a folder called **pychip8** on your machine and save our code below there with the file name **cpu.py**. We'll be using this folder for every module we create.



All being well, your pychip8 folder should look like this when you're finished.

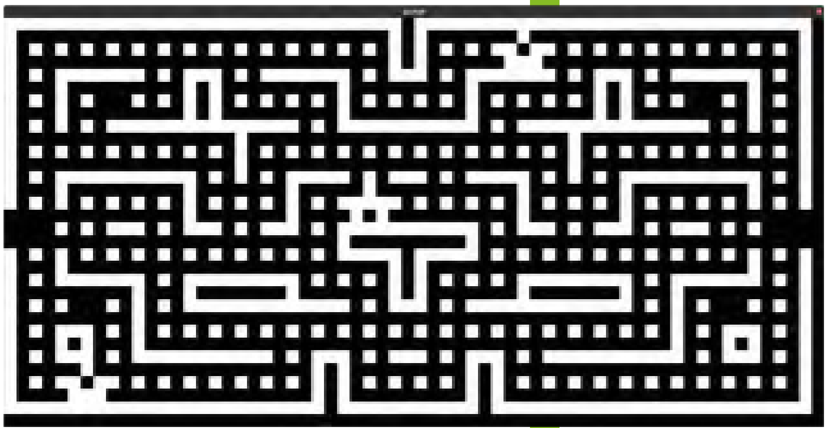
```
"this module defines the chip 8 cpu"

class Cpu:
    """this class represents the CHIP 8 cpu"""

    # game ram begins at address 0x200 / 512
    PROGRAM_START_ADDRESS = 0x200
    # the chip 8 works with 16 bit/2 byte opcodes
    WORD_SIZE_IN_BYTES = 2
```

```python
    # V[15/0xF] is used as a carry/no borrow flag for certain ops
    ARITHMETIC_FLAG_REGISTER_ADDRESS = 0xF
    FRAME_BUFFER_WIDTH = 64
    FRAME_BUFFER_HEIGHT = 32

    def __init__(self):
        # 4k of RAM
        self.ram = [0] * 4096
        self.program_counter = self.PROGRAM_START_ADDRESS

        self.index_register = 0
        self.general_purpose_registers = [0] * 16

        self.delay_timer = 0

        self.stack = []
        self.stack_pointer = 0

        self.keys = set()

        self.frame_buffer = [[bool()] * 32 for i in range(64)]

        self._load_font()

        self._current_word = 0
        self._current_operation = None
```

Can you see how we've defined the registers and buffers from the system's specification into simple Python types and collections? Don't worry about understanding all this code right now, as we'll be plugging in more parts of the computer as we go. This is just the 'heart' of the machine and where our core logic will run.

The CHIP-8, like almost all computers, works by following a sequence of instructions. We're going to load up files representing these instructions in a binary format (called ROMs) and process each instruction one by one in our CPU. The name for such an instruction is sometimes known as – you guessed it – an opcode!

The problem we face now is how can we get the data from a file into memory so our code can work with it? We can do this with a simple 'ROM loader' module that reads all the binary into a list



There are all kinds of simple games available for CHIP-8, from puzzlers to dinky *Space Invaders* clones. You'll find a good selection at **wfmag.cc/chip-roms**.

for us to use later. You can see how this works in the code below – save this into your folder with the file name **rom_loader.py**.

```python
import os

def get_rom_bytes(rom_name):
    """This method loads the bytes from a rom from the roms folder"""
    folder = os.path.dirname(os.path.realpath(__file__))
    rom_folder = os.path.join(folder, "roms")
    rom_file_path = os.path.join(rom_folder, rom_name)

    with open(rom_file_path, "rb") as file:
        file_bytes = file.read()
        return file_bytes
```

Here we're opening and reading the file in binary mode with the `"rb"` options string from the given file path and returning it from the `get_rom_bytes` method as a bytes object. This means we can then iterate over each byte in turn to decode each instruction for the CHIP-8.

How can we parse the raw binary from a file and turn that into instructions for the CPU? The CHIP-8 has a strange opcode definition, in that each opcode is a 16 bit / 2 byte value that contains the instruction *and* the data for that instruction. This would be simple enough were it not for the fact that apart from the first four bits, each following nibble can be part of an instruction or data – depending on the instruction in question. For instance, the opcode 0x00E0 means 'clear the screen', but opcode 0x1234 means 'jump to memory address 0x234'. A nibble might

"There are many emulators, but one of the most fun to start with is CHIP-8"

Wireframe

This tutorial first appeared in Wireframe, our sister magazine that lifts the lid on the world of video games. Every issue includes tutorials and in-depth interviews, along with news and reviews of the latest indie and triple-A games.

To find out more, visit their website at **wfmag.cc**.

Check out their subscription offers at **wfmag.cc/subscribe**.

```
"""
This module defines methods that define the different operations the CHIP-8
CPU could perform.
"""
import random
import font

# V[15] is used as a carry/no borrow flag for certain ops
CARRY_FLAG_ADDRESS = 0xF

def add_to_x(opcode, cpu):
    cpu.general_purpose_registers[opcode.x] += opcode.nn
    cpu.general_purpose_registers[opcode.x] &= 0xFF  # restrict the value to one
byte or less

def add_x_to_i(opcode, cpu):
    cpu.clear_arithmetic_flag()
    original_value = cpu.index_register
    result = cpu.index_register + cpu.general_purpose_registers[opcode.x]
    result &= 0xFFFF  # restrict the value to two bytes or less
```

Be sure to get the full CHIP-8 code from **wfmag.cc/wfmag54**, where you'll find the **operations.py** module, and more detailed comments that we couldn't fit in here.

sound strange, but it's just half a byte. A byte is 8 bits and can be represented with two hex digits, so 0xFF means 255. A nibble can be represented with one hex digit (e.g. 0xF for 15). This will come in handy later when we're manipulating our opcode data. One way to solve this problem is to abstract it into a class, which is exactly what we're going to do.

Our next piece of Python code is going to be an `Opcode` class, as shown below. Again, save this into your folder, this time with the file name **operation_code.py**.

"Don't worry if you don't understand all this code at once"

```
class Opcode:
    """This class represents the instructions and
data of an opcode"""

    def __init__(self, word):
        """
        This class takes in a 2 byte value/word
and parses the bytes
        to store them in different attributes for
later use

        Args:
            word: a 2 byte/16 bit value that
represents an opcode.
        """

        # We use bitwise AND with a mask to
extract specific nibbles.

        # a word should be no more than 16 bits
        self.word = word & 0xFFFF

        # we just want the most significant bits/
nibble
        # here so we bitshift right
        self.a = (word & 0xF000) >> 12

        self.nnn = word & 0x0FFF
        self.nn = word & 0x00FF
        self.n = word & 0x000F

        # Where don't use the lower nibbles,
bitshift
        # right to get just the raw value
        self.x = (word & 0x0F00) >> 8

        # Eg. we want 0x4 not 0x40
        self.y = (word & 0x00F0) >> 4
```

Don't worry if you don't understand all this code at once. The important thing to know is that we're parsing a 16-bit value and storing it in different fields of the class. `opcode.a` is the leftmost nibble, `x` the next one along, `y` the one after that, and `n` the very last. You can remember this with the hex-like pattern 0xAXYN. `opcode.word`, `opcode.nn`, and `opcode.nnn` are ways of accessing these nibbles in groups. This will come in handy later on.

Now we have a way to read the ROM and parse the contents of it, but what can we actually do with all this data? The CHIP-8 has 35 opcodes, but we shall only be using 33 of them. For simplicity, we're ignoring sound and any machine code operations. You can find a list of them on Wikipedia at **wfmag.cc/opcode**.

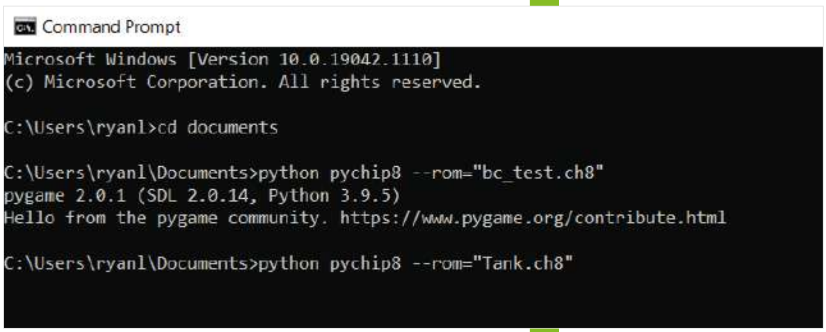
First, let's define a module that turns these opcodes into readable functions we can reference. Hex is fun, but it's not easy to understand what the raw numbers mean at times. The following bit of code is too long to run in full here, but you'll find it on our GitHub at **wfmag.cc/wfmag54**. It's the file named **operations.py** – here's a small snippet:

```
def add_to_x(opcode, cpu):
    pass

def add_x_to_i(opcode, cpu):
    pass
```

```
def add_y_to_x(opcode, cpu):
    pass
```

Don't worry that all these functions contain a `pass` statement at the moment; the important thing is that each one has the same signature and a clear name. Now we need to make another module, called **operation_mapping.py**, that lets us map the raw binary/opcode to one of the functions we've defined above. Modelling code without a full implementation like this is often known as stubbing, and can be a useful approach to working on a problem when you aren't entirely sure what the solution will be, but do need to have some idea of the interface you'll be using to solve it.

```
from operation_code import Opcode
import operations

def find_operation(word):
    opcode = Opcode(word)

    if word == 0x00E0:
        return operations.clear_display

    if word == 0x00EE:
        return operations.return_from_function

    if opcode.a == 0x1:
        return operations.goto

    if opcode.a == 0x2:
        return operations.call_function

    if opcode.a == 0x3:
        return operations.skip_if_equal

    if opcode.a == 0x4:
        return operations.skip_if_not_equal

    if opcode.a == 0x5:
        return operations.skip_if_x_y_equal

    if opcode.a == 0x6:
        return operations.set_x

    if opcode.a == 0x7:
        return operations.add_to_x

    if opcode.a == 0x8:
        if opcode.n == 0x0:
            return operations.set_x_to_y
        if opcode.n == 0x1:
            return operations.bitwise_or
        if opcode.n == 0x2:
            return operations.bitwise_and
        if opcode.n == 0x3:
            return operations.bitwise_xor
        if opcode.n == 0x4:
            return operations.add_y_to_x
        if opcode.n == 0x5:
            return operations.take_y_from_x
        if opcode.n == 0x6:
            return operations.shift_x_right
        if opcode.n == 0x7:
            return operations.take_x_from_y
        if opcode.n == 0xE:
            return operations.shift_x_left

    if opcode.a == 0x9:
        return operations.skip_if_x_y_not_equal

    if opcode.a == 0xA:
        return operations.set_i

    if opcode.a == 0xB:
        return operations.goto_plus

    if opcode.a == 0xC:
        return operations.generate_random

    if opcode.a == 0xD:
        return operations.draw_sprite

    if opcode.a == 0xE:
        if opcode.nn == 0x9E:
            return operations.skip_if_key_pressed
        if opcode.nn == 0xA1:
            return operations.skip_if_key_not_pressed

    if opcode.a == 0xF:
```



Once it's complete, you can run CHIP-8 straight from your command line, like this.

TECH SPECS

The CHIP-8 has the following specifications:

- 4kB of memory
- 16 general-purpose 8-bit registers (the 16th is also used as a special arithmetic flag at times)
- A 64 pixel-wide by 32 pixel-high frame buffer
- A stack that stores return addresses for function calls and nothing else
- Some way to store the pressed state of a 16-button keypad (0–F)
- An 8-bit index register
- An 8-bit program counter
- An 8-bit stack pointer
- An 8-bit delay timer register
- An 8-bit sound timer register (when above zero, a beep is made)

```
        if opcode.nn == 0x07:
            return operations.set_x_to_delay_timer
        if opcode.nn == 0x0A:
            return operations.wait_for_key_press
        if opcode.nn == 0x15:
            return operations.set_delay_timer
        if opcode.nn == 0x18:
            return operations.set_sound_timer
        if opcode.nn == 0x1E:
            return operations.add_x_to_i
        if opcode.nn == 0x29:
            return operations.load_character_
address
        if opcode.nn == 0x33:
            return operations.save_x_as_bcd
        if opcode.nn == 0x55:
            return operations.save_registers_zero_
to_x
        if opcode.nn == 0x65:
            return operations.load_registers_zero_
to_x

    raise KeyError(f"Opcode {word:#06x} not
present in list of valid operations")
```

Now we're cooking! This code will take a block of 16 bits we load from a ROM file, parse this into our custom `Opcode` class and use the properties of that class to figure out what corresponding functionality the opcode is referencing. If you refer to the table from **wfmag.cc/opcode**, you should be able to see exactly what our functions are going to actually do by looking up the related opcode.

Before we can wire this up further, we're going to need to add some extra functionality. There are a few opcodes that handle drawing to the screen, loading fonts, and input handling, so we need to define code that can manage this logic for us. Let's start with the internal font of the machine. The CHIP-8 has a single font that is hard-coded into memory, and usually stored in the first 512 bytes of memory (0x000–0x200). The font is basically a pattern of bits that describes the dots that make up each letter. Let's make a module with this pattern as shown below, and save it as **font.py**.

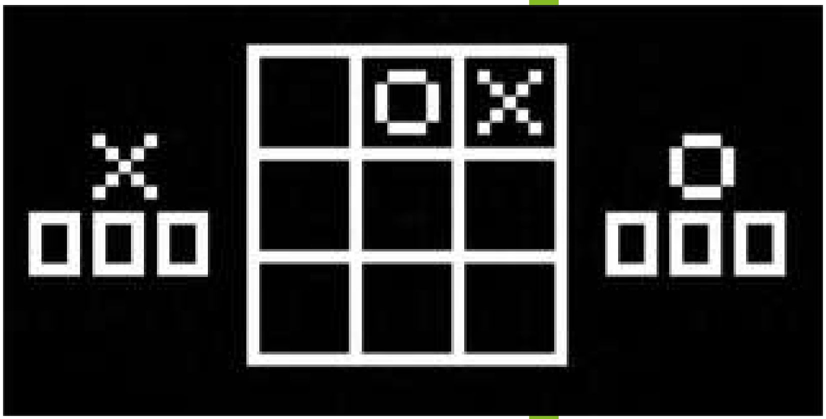
```
"""
Chars are 4x5. Each line must be a byte wide so
each line value is padded with 0x0.
Eg:
        0:
            ####  =  1111  =  F0
            #  #  =  1001  =  90
            #  #  =  1001  =  90
            #  #  =  1001  =  90
            ####  =  1111  =  F0
"""

# each char is 5 bytes long, so to get
# char 'B' you'd use DATA[CHAR_SIZE_IN_BYTES *
0xB]
CHAR_SIZE_IN_BYTES = 5

DATA = [0xF0, 0x90, 0x90, 0x90, 0xF0, # 0
        0x20, 0x60, 0x20, 0x20, 0x70, # 1
        0xF0, 0x10, 0xF0, 0x80, 0xF0, # 2
        0xF0, 0x10, 0xF0, 0x10, 0xF0, # 3
        0x90, 0x90, 0xF0, 0x10, 0x10, # 4
        0xF0, 0x80, 0xF0, 0x10, 0xF0, # 5
        0xF0, 0x80, 0xF0, 0x90, 0xF0, # 6
        0xF0, 0x10, 0x20, 0x40, 0x40, # 7
        0xF0, 0x90, 0xF0, 0x90, 0xF0, # 8
        0xF0, 0x90, 0xF0, 0x10, 0xF0, # 9
        0xF0, 0x90, 0xF0, 0x90, 0x90, # A
        0xE0, 0x90, 0xE0, 0x90, 0xE0, # B
        0xF0, 0x80, 0x80, 0x80, 0xF0, # C
        0xE0, 0x90, 0x90, 0x90, 0xE0, # D
        0xF0, 0x80, 0xF0, 0x80, 0xF0, # E
        0xF0, 0x80, 0xF0, 0x80, 0x80] # F
```

The code above is a layout of a series of nibbles in a list format that we can access, and each 5 bytes represents the raw pixels of a character. Each "1" equals a lit pixel for each character; each character is 4 bits across and consists of 5 rows. As we only use a nibble, only the most significant bits of each byte are used – that is why each value ends with 0.

Yes, it's a bit blocky, but it's a tiny, functioning version of *Space Invaders*. Pretty neat.

Next, we're going to need some way to handle input. The CHIP-8 had a very simple input mechanism – 16 keys from **0**–**F**. This fits exactly into a nibble. Let's define a module to map our keyboard keys to those values. Save the following as **keyboard_input.py**:

```
"""This module contains the keyboard input
handling logic"""

import sys
import pygame

keys = {}
keys[pygame.K_0] = 0x0
keys[pygame.K_1] = 0x1
keys[pygame.K_2] = 0x2
keys[pygame.K_3] = 0x3
keys[pygame.K_4] = 0x4
keys[pygame.K_5] = 0x5
keys[pygame.K_6] = 0x6
keys[pygame.K_7] = 0x7
keys[pygame.K_8] = 0x8
keys[pygame.K_9] = 0x9
keys[pygame.K_a] = 0xA
keys[pygame.K_b] = 0xB
keys[pygame.K_c] = 0xC
keys[pygame.K_d] = 0xD
keys[pygame.K_e] = 0xE
keys[pygame.K_f] = 0xF

def handle_input(cpu=None):
    """
    This function handles control input for this
program.
    """
    for event in pygame.event.get():

        # quit if user presses exit or closes the
window
        if event.type == pygame.QUIT:
            sys.exit()

        # check cpu registers and inject key input
        if cpu:
            if event.type == pygame.KEYDOWN:
                if event.key == pygame.K_ESCAPE:
                    sys.exit()

                if event.key in keys:
                    cpu.key_down(keys[event.key])

            if event.type == pygame.KEYUP:
                if event.key in keys:
                    cpu.key_up(keys[event.key])
```



CHIP-8 *Noughts and Crosses*. Perfect if you don't have a pen and piece of paper handy.

This code defines a dictionary to map keyboard input into and a function we can use to check which keys are pressed. We then pass that data into our CPU. Handy!

The last missing part of our emulator is the renderer. A renderer is simply some code that can take raw data and instructions and turn those into some form of visible output. We'll use Pygame Zero to draw to our screen and we can define our renderer like so, remembering that the CHIP-8 was limited to a 64×32 resolution with black and white output. The following needs to be saved with the file name **renderer.py**:

```
"""This module defines the renderer object and
related methods"""

import pygame

WHITE = pygame.Color(255, 255, 255)
BLACK = pygame.Color(0, 0, 0)

SCALE = 10

# The CHIP-8 display ran at only 64 * 32 pixels.
# this value scales the framebuffer
# so it's easier to view the emulator on modern
displays
screen = pygame.display.set_mode((640, 320))

def render(frame_buffer):
    """This method draws everything to the screen"""

    screen.fill(BLACK)

    for x in range(64):
        for y in range(32):
            if frame_buffer[x][y]:
```

Figure 1: If the **bc_test.ch8** ROM shows this message, then congratulations: you have a functioning CHIP-8 emulator, and you're ready to play some games!

```
                pygame.draw.rect(
                    screen,
                    WHITE,
                    (x *  SCALE, y * SCALE, SCALE,
SCALE))

    # Go ahead and update the screen with what
we've drawn.
    # This MUST happen after all the other drawing
commands.
    pygame.display.update()
```

The code above takes the frame buffer of the CPU and iterates through it, drawing a 10×10 white square for every pixel that should be set. This gives us a larger window to see the emulator running, and will be easier to use on a larger, modern display. A frame buffer is just a section of memory that stores things you want to show on a display monitor or TV. In our case, it's just a 2D list of Boolean values representing 'on' or 'off' for a pixel that we defined in our `Cpu` class earlier.

We're getting closer to having something work as all the core components of our system are now wired in. There's a bit more code we need to add to our CPU now we've defined all its dependencies. Let's go back and add some extra imports to the top of our CPU module, **cpu.py**:

```
from operation_code import Opcode
import operation_mapping
import font
```

Now append the code below to the end of the CPU module:

```
    def key_down(self, key):
        "This method sets a key as pressed"
        if key not in self.keys:
            self.keys.add(key)

    def key_up(self, key):
        "This method sets a key as released"
        if key in self.keys:
            self.keys.remove(key)

    def move_to_next_instruction(self):
        self.program_counter += Cpu.WORD_SIZE_IN_
BYTES

    def move_to_previous_instruction(self):
        self.program_counter -= Cpu.WORD_SIZE_IN_
BYTES

    def load_rom(self, rom_bytes):
        for i, byte_value in enumerate(rom_bytes):
            self.ram[Cpu.PROGRAM_START_ADDRESS +
i] = byte_value

    def set_arithmetic_flag(self):
        self.general_purpose_registers[self.
ARITHMETIC_FLAG_REGISTER_ADDRESS] = 1

    def clear_arithmetic_flag(self):
        self.general_purpose_registers[self.
ARITHMETIC_FLAG_REGISTER_ADDRESS] = 0

    def emulate_cycle(self):
        self._current_word = self.fetch_word()

        opcode = Opcode(self._current_word)
        self._current_operation = operation_
mapping.find_operation(self._current_word)

        self.move_to_next_instruction()
        self._current_operation(opcode, self)

    def fetch_word(self):
        word = self.ram[self.program_counter] << 8
| self.ram[self.program_counter + 1]

        return word

    def update_timers(self):
        if self.delay_timer > 0:
            self.delay_timer -= 1

    def _load_font(self):
        offset = 0x0
        for item in font.DATA:
            self.ram[offset] = item
            offset += 1
```

If you read through the code above, you should be able to understand what each method is doing

as there is a comment in each one to help you understand the code. We'll need this for the next step as we're going to revisit our `operation` class and fill in the blanks we left earlier. This is going to be quite a lot of code – as before, it's too much to print here, so download the file **operations.py** from our GitHub at **wfmag.cc/wfmag54**. Don't worry too much about what each method does; focus more on what opcode it relates to and what functionality it achieves for our emulator. There are several comments to help you understand each opcode, if you're interested.

We only need to write one more module to get our emulator working and to see our hard work pay off. We need an entry point to our code that initialises and runs it in the right order. Let's go ahead and create a file called **__main__.py** (NB: there are two underscores either side of the word 'main') that will handle all this for us.

```
"""This is the main entry point for the program"""

import argparse
import keyboard_input
import renderer
import rom_loader
import pygame

from cpu import Cpu

if __name__ == "__main__":

    parser = argparse.ArgumentParser()
    parser.add_argument("-r", "--rom",
required=True, type=str, help="the name of the rom
to run in the emulator")
    args = parser.parse_args()

    cpu = Cpu()
    clock = pygame.time.Clock()
    rom_bytes = rom_loader.get_rom_bytes(args.rom)
    cpu.load_rom(rom_bytes)

    pygame.display.set_caption("pychip8")
    pygame.init()

    # main loop
    while True:
        keyboard_input.handle_input(cpu)

        # The CHIP-8 is reported to run best at
around 500 hz
        # The update loop runs at 60 fps. 60 * 8 =
480, which is close enough.
        for _ in range(8):
            cpu.emulate_cycle()

        cpu.update_timers()
        renderer.render(cpu.frame_buffer)

        # delay until next frame.
        clock.tick(60)
```

The code above sets up all the dependencies, loads the ROM data, and runs the core loop of the CHIP-8 CPU, which will read and evaluate instruction by instruction. Don't worry about "running out", as almost every ROM loops; they're not designed to terminate without user input.

The last thing we'll need is a ROM to test this out with. First, let's use a debug ROM – the best I've found is called **bc_test** created by BestCoder. Create a folder called **roms** in your working directory and copy **bc_test.ch8** into it from this repository: **wfmag.cc/chiptest**.

With the ROM in place, open the directory that contains your **pychip8** folder in your terminal of choice and run the command `python pychip8 --rom="bc_test.ch8"`. You should see something like the output in **Figure 1**. If you do, have a round of applause – you have a working emulator! If you see nothing on the screen, the error's likely in the renderer module; if you see a screen like the one in **Figure 2**, then you'll be able to work out what's gone wrong by checking the error codes at **wfmag.cc/error-codes**.

Once you're sure the test passes, it's time for real games! There are plenty of ROMs available for the CHIP-8 – you'll find several at **wfmag.cc/chip-roms**. Just copy the files into your **roms** folder and run the command, replacing the test ROM name with the name of the ROM you want to run, and bingo: you're playing games on a computer that you wrote yourself!

Figure 2: If running the **bc_test.ch8** ROM gets you a message like this, then you can troubleshoot your emulator with the error list at **wfmag.cc/error-codes**.

Raspberry Pi Operating Systems

Whatever you want to do with Raspberry Pi, there's an operating system for that, as **Phil King** finds out

Based on Debian, a flavour of Linux, the official Raspberry Pi OS is an excellent, easy-to-use operating system, but it's not the only show in town.

You might want to give an alternative general-purpose OS a go, such as the popular Ubuntu with its slick interface and large software catalogue, or the user-friendly and accessible Manjaro. If you're feeling adventurous, there's also RISC OS: an evolution of the Acorn Archimedes operating system dating back to 1987, it's super-fast and offers a fascinating insight into how desktop OSes used to look and work.

Other Raspberry Pi operating systems are aimed at particular use cases. For media playback, there's LibreELEC and OSMC: both based on Kodi, they offer a slick interface for navigating your music and video collections, as well as streaming media. Retro gaming fans are also well catered for with a choice of RetroPie, Recalbox, and Lakka for emulating old consoles and computers. Then there are specialist OSes such as OctoPi (3D printing) and Home Assistant (smart home).

Trying out a new operating system is as easy as writing it to a microSD card with the Raspberry Pi Imager tool (**magpi.cc/imager**), which has many OS options already built-in, so why not give it a go?

Ubuntu Desktop

- Type: General-purpose
- Available: Raspberry Pi Imager, ubuntu.com/raspberry-pi
- Works on: Raspberry Pi 4 (4GB+), 400, CM4 (4GB+)
- Minimum card size: 16GB

People have been using Raspberry Pi as a cost-effective PC alternative for years, but you don't have to stick to the official Raspberry Pi OS: Ubuntu is a popular flavour of Linux used on millions of PCs around the world.

While Ubuntu is also available in more lightweight Server and Core versions, the new Desktop edition is the fully-fledged version of the Linux distro. So if you've used it before, it'll all be very familiar in how it looks and works.

Available in the menu of the Raspberry Pi Imager tool (**magpi.cc/imager**), it's a pretty large download and takes a little while to write to a 16GB or larger microSD card. Boot it up on Raspberry Pi and it starts with a system configuration welcome wizard where you can set up the keyboard type, WiFi connection, and add a new user. It then takes a few minutes to configure everything.

The GNOME desktop environment is aesthetically pleasing with an applications sidebar, launcher, and search facility. The downside is that it's rather sluggish to use: there's a slight delay whenever you open or do anything on the desktop. Not everything works out of the box, either: you may well get a message saying you need to install some extra packages to do something such as playing MP4 videos.

One of the main advantages of Ubuntu is its huge software library. Pre-installed apps include the Firefox web browser, Thunderbird email client, Rhythmbox music player, and LibreOffice suite. More can be installed via the Ubuntu Software tool – or, a little confusingly, also from the Snap Store, Synaptic Package Manger, or Terminal. You may well find more recent versions of some software than in Raspberry Pi OS, such as Python 3.8. The GPIO pins can be used by installing the LGPIO Python library.

If you want to stay on the cutting edge, Ubuntu gets a major update twice a year, although this can sometimes cause issues with stability.

- Fully-fledged desktop OS
- Slick, modern GUIL
- Large software catalogue
- Can be a bit sluggish and flaky

RISC OS Pi

- Type: General-purpose
- Available: Raspberry Pi Imager
- Works on: All models
- Minimum card size: 2GB

"It'll even run on the original Model B"

Now this one takes us back. RISC OS was first developed by Acorn to run on its 32-bit ARM-based Archimedes home computer in 1987. Since Raspberry Pi also uses an ARM CPU, it seems only fitting that the OS is available for it. A community of enthusiasts continues to develop RISC OS, including a special version tailored for Raspberry Pi 400 (available from **riscosopen.org**).

The first thing you'll notice when installing RISC OS is how compact it is. This is an ultra-lightweight OS that only requires a 2GB microSD card – there's no advantage in using a larger one, since the OS can only use the first 2GB. Amazingly, it'll also work on any Raspberry Pi model, even the original Model B from 2012.

The old-school GUI is a world away from Raspberry Pi OS or Ubuntu. It looks very retro and applications are grouped together in folders rather than in a main menu. The other aspect you'll notice is that it runs like lightning, even though it only operates on a single core of the CPU.

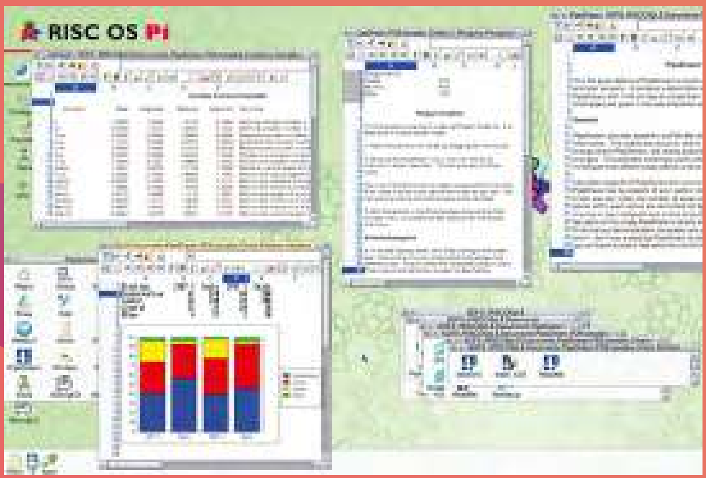
An intriguing aspect is that it's based around using a three-button mouse: the left button is select, middle (or scroll wheel) displays a contextual menu, while right is 'adjust' (a modified version of select). Once you get the hang of it, this can prove very effective. If you only have a two-button mouse, you can always assign a keyboard key as the third button.

Bundled software includes the primitive NetSurf web browser, PipeDream office suite, and a selection of basic games (called 'diversions'). Using the StrongEd text editor, it's possible to program in languages such as BBC BASIC, Lua, and Python. Extra software can be installed using the !Store and PackMan package managers.

The biggest downside is that Raspberry Pi's on-board wireless LAN is not yet supported, so you'll need to use a wired Ethernet connection or a special WiFi HAT to get online. In addition, the GPIO pins on more recent Raspberry Pi 4 models aren't recognised, while the USB 3.0 ports only work at USB 2.0 speeds.

Still, it's well worth a look for the nostalgic and the curious and could prove useful for learning how computers work under the bonnet.

- **Ultra-lightweight OS**
- **Extremely fast to use**
- **Looks very old-fashioned**
- **No support for on-board WLAN**



Manjaro ARM Linux

- Type: General-purpose
- Available: Raspberry Pi Imager, manjaro.org
- Works on: Raspberry Pi 3, 4, 400, CM3/3+/4
- Minimum card size: 8GB

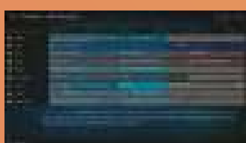
Manjaro aims to be a simpler, more user-friendly version of Arch Linux while providing all the benefits of cutting-edge software – it boasts its own large repository alongside the Arch User Repository.

You can install Manjaro with one of four desktops – KDE Plasma, GNOME, Xfce, MATE – or the lightweight Sway window tiling manager, or a minimal version with no desktop.

- Based on Arch Linux
- Highly customisable
- Good choice of desktops
- Large software repository

LibreELEC

- Type: Media centre
- Available: Raspberry Pi Imager, libreelec.tv
- Works on: All models
- Minimum card size: 16GB



Based around the Kodi media player software, LibreELEC is a lightweight single-purpose distro that's simple to set up and use. The menu system is slick and easy to navigate, with sections including movies, TV shows, and music. You can also stream media by installing add-ons for services such as YouTube, BBC iPlayer, and Disney+. With a Raspberry TV HAT attached, you can even use it as a personal video recorder.

- Slick, easy-to-use interface
- Supports most media files
- Highly customisable with add-on
- Watch and record live TV (with TV HAT)

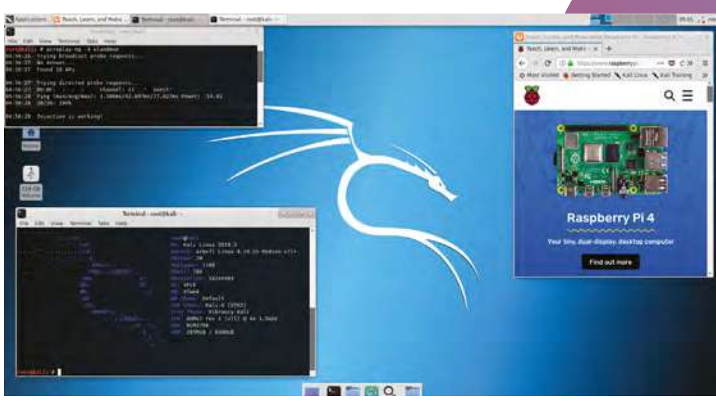
Kali Linux

- Type: Security testing
- Available: kali.org
- Works on: All models
- Minimum card size: 16GB

Kali is a specialist distro for information security tasks such as penetration testing. Once installed on Raspberry Pi, you can use it to audit the security of other devices on your network.

There's no configuration wizard, so you'll need to set up a few things manually. You do get a desktop (Xfce), though, and a set of built-in security apps, plus the Firefox web browser.

- Specialist security distro
- Good range of OS versions
- Set of built-in testing tools
- Some manual configuration needed



OSMC

- Type: Media centre
- Available: Raspberry Pi Imager, osmc.tv
- Works on: All models
- Minimum card size: 16GB

Based around the Kodi media player, OSMC is a fully-fledged OS, enabling you to install other applications. So it may prove more versatile if you need to also use your Raspberry Pi for anything else, such as a web server.

When using it for Kodi media playback, OSMC works very similarly to LibreELEC with an easy-to-use menu system to navigate media categories such as movies, TV, and music.

- Fully-fledged operating system
- Versions for different models
- Kodi media playback
- Versatile and customisable

RetroPie

- Type: Retro gaming
- Available: Raspberry Pi Imager, retropie.org.uk
- Works on: All models
- Minimum card size: 8GB

Built on top of Raspberry Pi OS, RetroPie brings together a suite of emulators for over 50 retro consoles and computers. Running it on a Raspberry Pi 4 or 400 gives you the most power, enabling better emulation of later systems such as the PlayStation 2, GameCube, and even the Wii.

Use it in a standard setup with a monitor or, for a more authentic experience, a bartop or full-size arcade system.

- Emulates over 50 systems
- Works on all models
- Customisable menus and settings
- Large community to offer advice

Raspberry Pi OS versions

The official operating system for Raspberry Pi is based on Debian Linux. There are several versions available...

With Desktop (32-bit): The default option in Raspberry Pi Imager, it features the Pixel GUI and includes core applications such as the Chromium web browser and VLC media player.

Lite: This is a minimal version of the OS with no desktop installed, just a command-line interface. Quick to boot, it's ideal for a headless setup, typically on a Raspberry Pi Zero.

Full: Identical to the default option, with a GUI, but with additional recommended software pre-installed such as the LibreOffice suite and various programming IDEs.

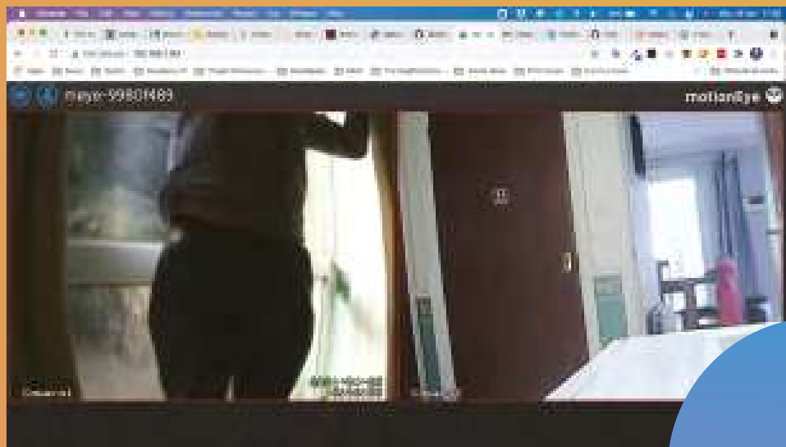
64-bit: Not yet available in Raspberry Pi Imager, this is a beta test edition that only works on 64-bit models (3, 4, 400, or CM4) to unleash their full power. Download it from **magpi.cc/rpios64**.

Lakka

- Type: Retro gaming
- Available: lakka.tv
- Works on: All models
- Minimum card size: 8GB

Lakka is another solid option for retro gaming. Each game is implemented as a libretro 'core', while RetroArch takes care of the inputs and display. So, while it doesn't support quite so many systems as its rivals, it offers a single, unified, and consistent configuration for all its emulators. Controllers typically work as soon as you plug them in, too.

- Versions for all models
- Download from website
- Painless configuration
- Samba sharing for ROMs



MotionEyeOS

- Type: Security camera
- Available: magpi.cc/motioneyeos
- Works on: All models
- Minimum card size: 1GB

This OS turns your Raspberry Pi into a video surveillance system using a Camera Module and/or USB camera. You can even set up a network to cover different rooms of your home, monitoring all the camera feeds in a web interface. It can be set to capture photos and video whenever motion is detected on a camera, and send you email/push alerts.

- CCTV security system
- Options for multiple cameras
- Can monitor several rooms
- Download from website

Recalbox

Recalbox offers a wide range wide of emulators for retro gaming – if not quite so many as RetroPie. It does come with the Kodi media player built-in, though, so it's ideal if you're look for an all-in-one entertainment system.

Recalbox also adds a share partition on the microSD card so you can easily remove it from Raspberry Pi to add game ROMs on another computer.

- Type: Retro gaming
- Available: Raspberry Pi Imager, recalbox.com
- Works on: All models
- Minimum card size: 8GB

- Versions for every model
- Wide range of emulators
- Kodi media player built-in
- Share partition for ROMs

info-beamer

- Type: Digital signage
- Available: Raspberry Pi Imager, info-beamer.com
- Works on: Raspberry Pi 2, 3, 4, Zero W, CM3/3+/4
- Minimum card size: 8GB

Digital signage installations are typically found in shops, restaurants, cinemas, museums, etc. To create your own, all you need is a Raspberry Pi with info-streamer installed and a monitor or TV.

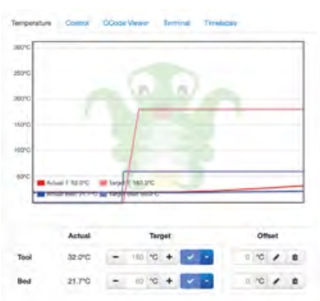
Sign up for a free account and you can monitor and control your device(s) from a web dashboard. With multiple screens and Raspberry Pi boards, you can even set up a video wall.

- Digital signage solution
- Cloud dashboard
- Monitor multiple devices
- Build your own video wall

OctoPi

- Type: 3D printing
- Available: Raspberry Pi Imager, octoprint.org
- Works on: Raspberry Pi 3, 4, 400, CM3/3+/4
- Minimum card size: 8GB

OctoPi contains the OctoPrint software for monitoring and controlling a 3D printer remotely from a web browser. It adds WiFi to any 3D printer with a USB port, for easy file dropping and remote control via a web dashboard. You can also add a camera to the setup for a live video stream of the 3D printing process, and to record time-lapse videos.



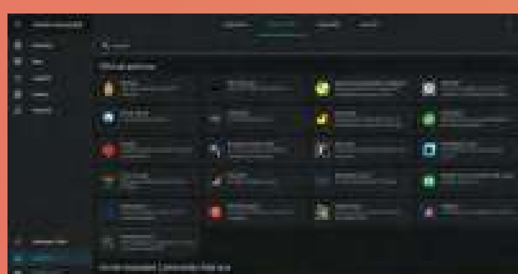
- Control a 3D printer
- Access via web interface
- Live camera stream
- CuraEngine slicing

Home Assistant

- Type: Smart home
- Available: Raspberry Pi Imager, home-assistant.io
- Works on: Raspberry Pi 3, 4, 400
- Minimum card size: 4GB

Home Assistant is an open-source ecosystem for smart home automation and boasts a helpful worldwide community of enthusiasts. Easy to set up and use, it features 'Integrations' (plug-in modules) to support over 1,000 smart devices; you can even program your own in Python. Little wonder PJ Evans opted to use it for his tutorial series on home automation starting in *The MagPi* issue 99 (**magpi.cc/99**).

- Over 1,000 devices supported
- Sophisticated web dashboard
- Powerful automations
- Google or Alexa voice control



Homebridge

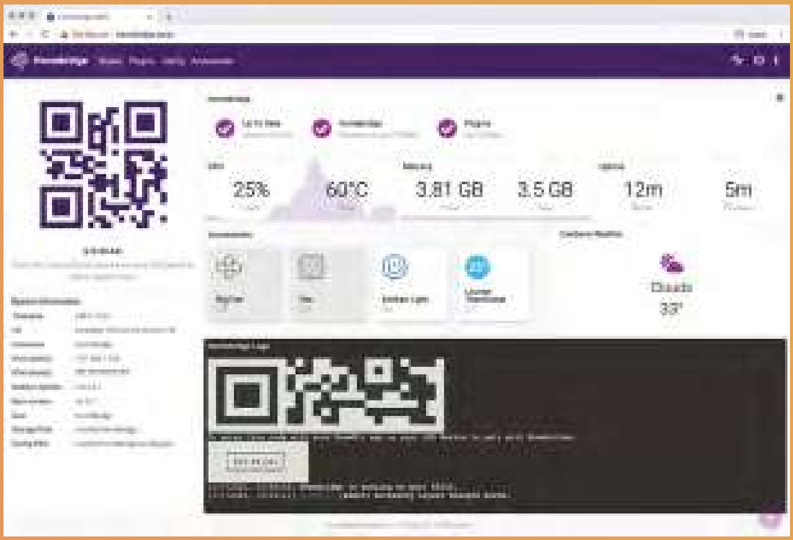
- Type: Smart home
- Available: Raspberry Pi Imager, homebridge.io
- Works on: All models
- Minimum card size: 4GB

Apple's HomeKit is great for controlling your smart home, but it's not compatible with all devices. Homebridge offers a Raspberry Pi-based solution, enabling you to integrate other devices such as Nest thermostats, Ring video doorbells, Philips Hue light-bulbs, and Sonos speakers.

Your Homebridge setup can be controlled via a web-based interface, including installing or removing plug-ins – there are over 2,000 available.

- Use other devices with HomeKit
- Over 2,000 plug-ins
- Web-based interface
- Voice control with Siri

"Your Homebridge setup can be controlled via a web-based interface"



Other OSes

We didn't have space in this feature to cover every available operating system for Raspberry Pi. Here are a few others to consider…

Alpine Linux (alpinelinux.org): Lightweight OS with an emphasis on security.

ALT (altlinux.org): Russian language OS, available in Raspberry Pi Imager under 'Other language-specific OS'.

Arch Linux ARM (archlinuxarm.org): Lightweight and flexible OS, the basis of Manjaro.

CentOS (centos.org): Robust Linux distro, popular for running web servers.

Chromium OS (magpi.cc/chromiumos): The open-source basis for Chrome OS, as used in Chromebooks.

DietPi (dietpi.com): Extremely lightweight Debian-based OS with a simple interface.

emteria.OS (emteria.com): Full build of Android, aimed towards industry use.

Fedora ARM (arm.fedoraproject.org): ARM version of Fedora Linux, only works on Raspberry Pi 2 and 3.

FreeBSD (freebsd.org): UNIX-like operating system, an alternative to Linux.

FydeOS (fydeos.io): Customised version of Chromium OS for Raspberry Pi 4 and 400.

Gentoo (gentoo.org): Highly customisable Linux distro, not easy to install.

Lubuntu (lubuntu.net): Lightweight Linux OS based on Debian, with minimal LXDE desktop.

NetBSD (netbsd.org): Fast and secure UNIX-like OS based on BSD.

piCorePlayer (picoreplayer.org): Minimal OS designed for remote music playback.

Pi MusicBox (pimusicbox.com): Stream music or play locally stored tracks and podcasts.

Slackware ARM (sarpi.penthux.net): Slackware is one of the oldest Linux distros.

SliTaz (slitaz.org): Compact OS that works completely in memory from an attached USB stick.

TLXOS (thinlinx.com): A 'thin client', available as a 30-day trial in Raspberry Pi Imager, it's a compact OS that's useful for kiosk displays.

Void Linux (voidlinux.org): Independent general-purpose Linux distro.

Hiwonder **TonyPi**

SPECS

DIMENSIONS:
186 × 113 × 373 mm; 1.7 kg

COMPONENTS:
Raspberry Pi 4 4GB, 11 V battery, 3MP camera (480×800 pixel) 14 × LX-824HV servos, on-board lithium-ion battery pack with DC charging point (US adapter), power level indicator

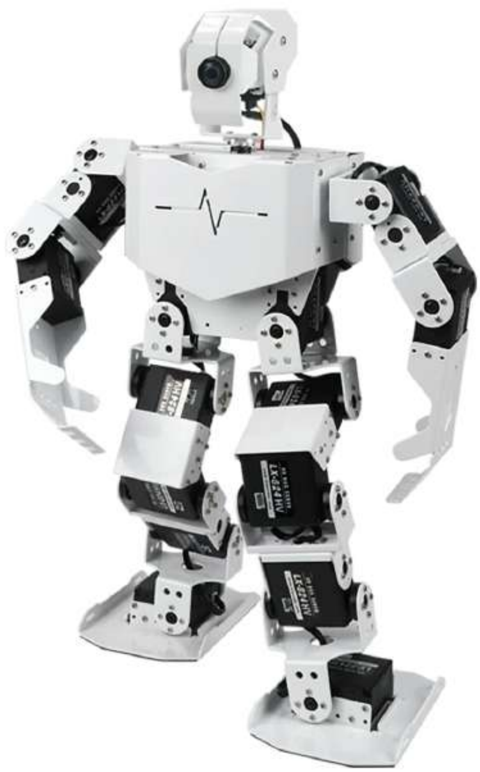
ACCESSORIES:
3 × Tag cards, 3 × colour balls, servo wire, screwdriver

▶ Hiwonder ▶ **hiwonder.hk** ▶ £380 / $519

Raspberry Pi 4-powered humanoid robot with built-in smarts and options to set up bespoke behaviours. By **Rosie Hattersley**

TonyPi is a 37 cm-tall humanoid robot with plenty of personality to keep you engaged as you learn fun ways to interact with him. He can either be controlled via an iOS or Android app, or you can use a Raspberry Pi, Windows PC, or Mac.

Containing a Raspberry Pi 4, TonyPi comes preassembled (aside from the connectable head),

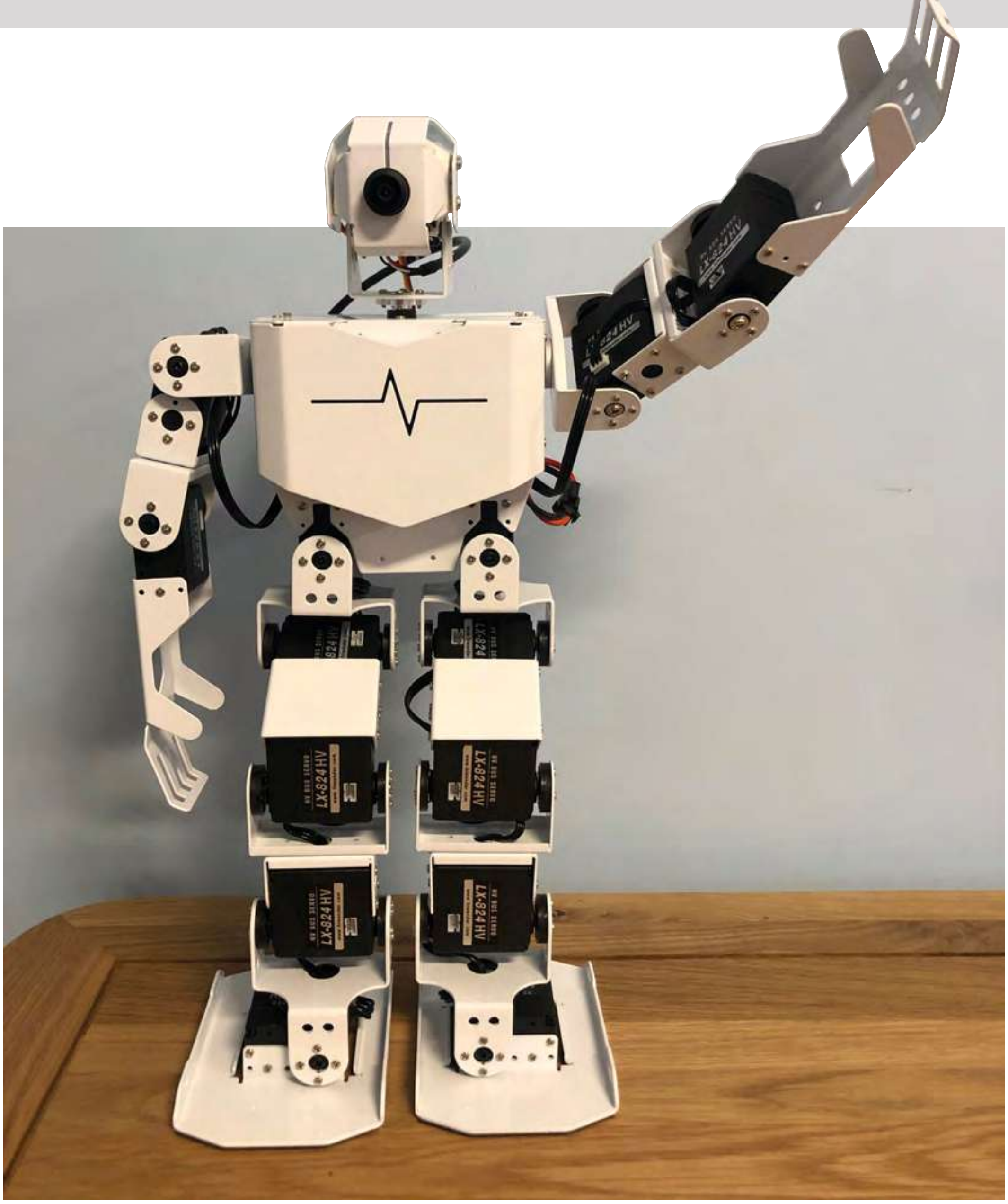


▶ With 14 servos, TonyPi is both powerful and responsive to user commands

so the fun here is all about controlling him. He's a solidly built fella, with angular joints and broad lower arms and hands primed for scooping up coloured balls on command. We rather liked his all-white finish, which contrasts with his 14 deliberately visible LX-824HV servos. Agility-wise, TonyPi can walk, shimmy, wave, bow, get up, and turn his head – all from the smartphone or tablet app, while more advanced controls can be added as custom list items via Python and OpenCV.

> We tried to fool him with posters of people and various charming but inanimate objects, but TonyPi was unmoved

Swiping a finger in the window section of the Hiwonder app moves TonyPi's head. You'll need to swipe several times to get TonyPi to move much, though. Body movements can be initiated using the step controls at the top of the on-screen control panel, or by tapping within the circular control area. The upper control section causes TonyPi to step forwards and backwards and turn gently to one side, while the circular control section sees him jump peremptorily whichever way he's instructed.



TonyPi is a great-looking robot that can be easily controlled via Android or iOS apps

Clear vision

TonyPi's object recognition ability is great. One neat trick is his ability to recognise faces, waving in response. We tried to fool him with posters of people and various charming but inanimate objects, but TonyPi was unmoved.

'Colour tracking' was also impressive: select a colour and move the corresponding ball gently in front of his 3-megapixel camera and TonyPi's head will follow the ball's movements. He will also nod when shown an object that's red, green, or blue – our orangey-red sofa was an easy win – but he doesn't always get it right.

Line following was largely successful: TonyPi strode along confidently on straight stretches, slowing but still following the line round gentle bends. It's best to set his head to point down so he is able to follow the trail; if not, he may veer off unexpectedly at an angle. Three QR codes are included in the box: place these against a light background and TonyPi will wave, shimmy, or shuffle his feet. You can also create your own QR codes and have your robot do something else, having programmed a different action in Python using the instructions provided.

Be aware that switching user modes causes TonyPi to revert to his initial position, as we inadvertently discovered. If TonyPi is lying down, you're likely to have one heck of a shock when you switch him back on and tap the 'get up' instruction on the Hiwonder app. TonyPi launched himself backwards as he automatically tried to stand upright from the prone position we'd left him in. Luckily, your reviewer caught him just as he was about to disappear off the edge of the kitchen table.

Instructional PDFs really help in getting the most from using TonyPi, but using a PC or Raspberry Pi as a control via SSH and a smartphone hotspot is extremely fiddly, more than pi-top's Robotic Kit with its built-in IP address-displaying OLED [**magpi.cc/108**]. Once set up, though, possibilities such as adding a microphone and trying voice recognition come into play.

Verdict

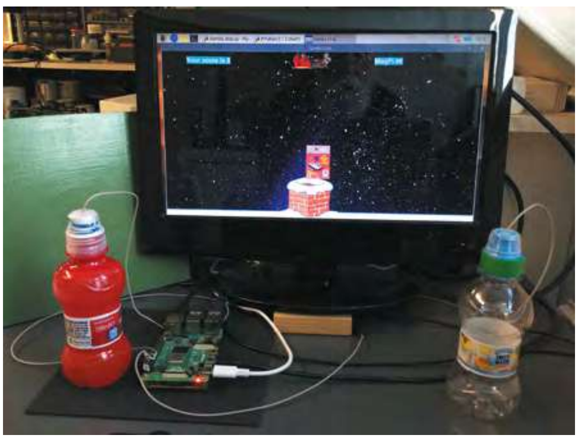
TonyPi is an impressively designed robot with solid object recognition features. For longer-term entertainment, be prepared to log in remotely and use Python to access his more advanced features.

8/10

10 Amazing: Mike Cook projects

A look back at some of Mike Cook's best builds in The MagPi

You may have read last month that Mike Cook, legendary tech writer and author of our monthly Pi Bakery since issue 33, has decided to retire from regular tutorials in *The MagPi*. While we'll see him again in the future, we wanted to highlight ten very cool things that he made over nearly 80 issues. Thank you, Mike!

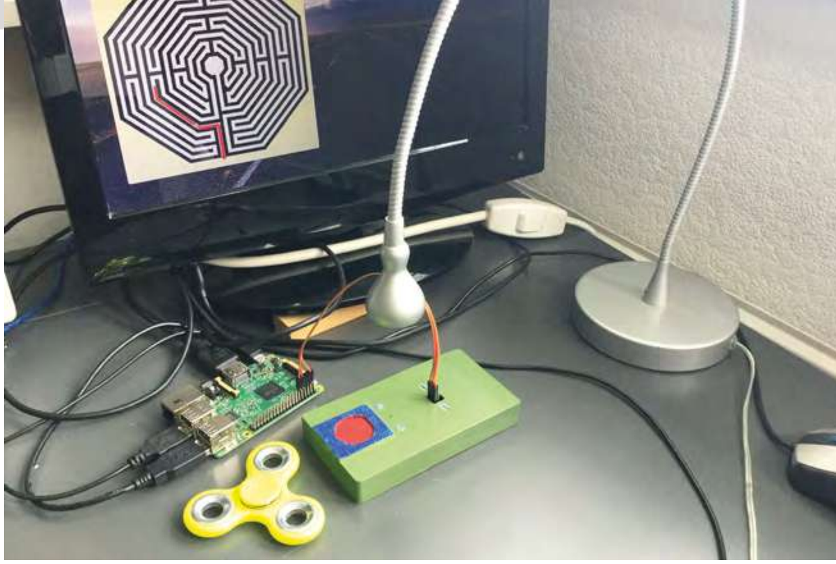


▲ Santa's Drop

Deliver *The MagPi* magazines festively

A fun holiday game to make and play, where you take the reins of Santa's sleigh to deliver *The MagPi* magazines to kids everywhere. Unfortunately, the real Santa is too expensive to use monthly.

magpi.cc/101



▲ Fidget Spinner Maze Runner

Spin to escape

This project tests your fidget-spinning skills to see just how fast you can get out of the maze determined by how fast you spin.

magpi.cc/60

▲ Danse Macabre

A skeleton dance

Perfect for the spooky season, these scary skeletons can be made to deviously dance by your own hands, while classical music is played of course.

magpi.cc/50

▼ Cut-out theatre

Interactive story time

Use a tablet to control this wonderful, old-school storytelling method that will be sure to impress and entertain many kids, young and old.

magpi.cc/57

▼ Trill guitar

Mini MIDI guitar

This guitar uses a mixture of Raspberry Pi Pico and Trill touch sensors to create a very customisable MIDI guitar. We love the case for it as well.

magpi.cc/102

▲ The Large Head 'N Ron Collider

Halloween games

Meet Head and Ron, glowing spooky creatures that are used as part of the Collider game, where you try and match the lights at the right moment.

magpi.cc/62

▲ Raise a Glitch Storm

Noise corruption

Using some rotary encoders, you can interact with an oscilloscope to create weird and wonderful noises, for fun and scary movies.

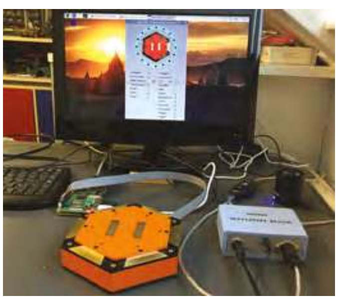
magpi.cc/95

▸ Hex-a-Pad

Six-sided sounds

This cool little instrument is one of many that Mike has made, with sides you can tap for sound. This two-part tutorial starts in issue 90.

magpi.cc/90

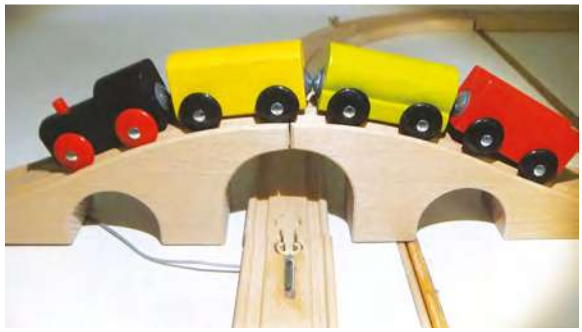


▾ The Story Train

Trolley problem solved

Wooden train sets are irresistible. You have to play with them. Linking them up to an e-book allows you to turn pages using the train, a great way to keep kids engaged with reading.

magpi.cc/41



▾ Pico-Voice

Voice modulation

Use various knobs and dials to change your voice using this Raspberry Pi Pico project in a very cool box.

magpi.cc/106

RASPBERRY PI PROJECTS FOR DUMMIES

Mike is also the author of *Raspberry Pi Projects for Dummies*, which you can grab at **magpi.cc/mikecook**.

Learn circuit design with Raspberry Pi

Discover how to design your own electronic circuits with these resources. By **Phil King**

Getting Started in Electronics

AUTHOR Forrest M Mims

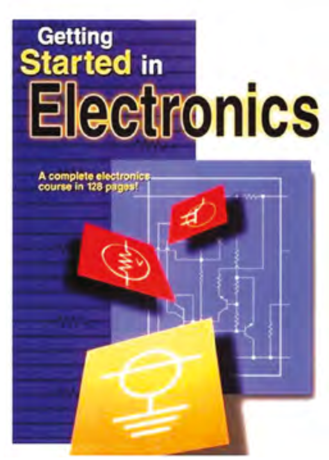
Price: £24.99 / $20

magpi.cc/electronicsmims

Who better to teach you the fundamentals of electronic circuits than Forrest M Mims, the designer of the original Altair 8800 home computer? Billed as a complete electronics course in 128 pages, this classic tome is all the more remarkable for its handwritten text and hand-drawn diagrams! While there are more in-depth electronics books around, this one will certainly give you a solid grounding.

Starting with the basics of electricity, it takes you on a tour of digital and analogue components and how they work. It moves on to integrated circuits and digital logic gates using switches and transformers. Linear circuits are also covered; responding to a wide range of voltages, they have many applications.

Of most interest to us is the section on using electronic components to make temporary



(i.e. on a breadboard) and permanent circuits. There are 100 tried and tested circuit designs to explore, arranged into categories such as basic, photonic, digital, and linear.

Online courses

Study circuit design with these interactive courses

CRASH COURSE ELECTRONICS & PCB DESIGN

This Udemy course is aimed at those learning electronics from the ground up who wish to leverage that knowledge to build actual PCBs.

▶ magpi.cc/crashcoursepcb

TINKERING FUNDAMENTALS: CIRCUITS

Participation is essential in Coursera's hands-on workshop as students are encouraged to try things out, ask questions, and share ideas with each other as a community.

▶ magpi.cc/tinkeringcircuits

ADVANCED DIPLOMA IN BASIC ELECTRONICS

This free Alison course covers the principles of electronics and some of the theorems guiding the design of electrical circuits.

▶ magpi.cc/alisonelectronics

An Introduction to Electronics

AUTHOR
Open University

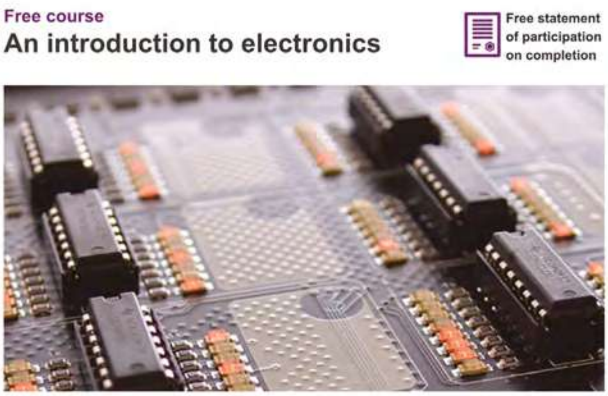
Price:
Free

magpi.cc/openlearnelectronics

This OpenLearn course is an adapted extract from the Open University course T212 Electronics: sensing, logic and actuation. Just sign up for a free account to enrol. Taking around ten hours at your own pace, it starts with the basic theory of electrical circuits, including Ohm's law, Kirschoff's laws, and series and parallel circuits.

It then goes on to cover some fundamental circuits – including voltage dividers, the Wheatstone bridge, operational amplifiers, and sensors – before exploring the concept of signal processing using sounds.

Each section features videos and/or interactive elements, By the end of the course, you should be able to use basic components to design simple electronic circuits, as well as having learnt a lot more about the general concepts involved.



HackSpace Magazine #48

AUTHOR
Ben Everard

Price:
£6 / $8 (or free PDF)

hsmag.cc/issue48

The ultimate achievement in circuit design is to create your own printed circuit board (PCB), but it may seem a daunting prospect. The latest issue of our sibling magazine, HackSpace, demystifies the process with a 12-page feature showing you how to get started.

As the article explains, "If you can build electronics on a breadboard, you can design a PCB. There's no magical difference. Instead of wires, you just have traces of copper."

Using the EasyEDA online software (**easyeda.com/editor**), it takes you through all the steps of designing a PCB to work with a Raspberry Pi Pico. Starting with a schematic, various components are added, along with custom artwork and copper traces for connections. Finally, you can send your board design off to be manufactured.

Books to read

Check out these info-packed books

SIMPLE ELECTRONICS WITH GPIO ZERO

Build breadboard-based electronic circuits connected to Raspberry Pi and control them with Python programs and the easy-to-use GPIO Zero library.

▶ magpi.cc/gpioessentials

MAKE: ELECTRONICS

A great primer for general electronics, it's full of interesting facts and runs you through 36 'experiments' building circuits using standard components.

▶ magpi.cc/makeelectronics

ELECTRONIC CIRCUIT DESIGN IDEAS

Aimed at the more advanced user, this weighty compendium covers over 170 circuit types and explains them in great detail with diagrams.

▶ magpi.cc/circuitdesignideas

Akkie

Can you be a maker without knowing electronics? Akkie was before Raspberry Pi

- Name **Akira Ouchi** | Occupation **Infrastructure engineer**
- Community role **Maker** | Website **github.com/Akkiesoft**

"I had no electronic skills at all before I learned and started playing with Raspberry Pi."

Not the kind of thing you'd expect to hear from one of the best-known members of Japan's wonderful maker community, Akira Ouchi, or Akkie as he's best known. He had been doing little projects though, like sticking a pen into an old mouse to create a sort of tablet, or making use of an optical disc drive to look after his hamster.

"Although the term 'IoT' was not common in 2009, I combined the optical drive, cardboard, and springs to create a thing that could feed the hamster or control the power switch of the air conditioner by [moving] the tray of the optical drive," Akkie tells us. "It is quieter, easier, and smarter to implement similar functions with a Raspberry Pi, server monitor, and infrared LED nowadays."

This optical drive would take care of Akkie's hamster while he was away

When did you learn about Raspberry Pi?

I discovered Raspberry Pi on the RS Components' website on 12 March 2012 – I found this from my Twitter [Ed, see photo]. At that time I used a Linux netbook for my optical drive IoT project to demonstrate at some events. I thought Raspberry Pi could make it compact the demo stuff. After replacing the netbook with Raspberry Pi, I created a wearable optical drive attached on the top of a helmet, and Raspberry Pi built-in optical drive.

What are your favourite things that you've made with Raspberry Pi?

My fave is a Plarail (Japanese train toy for kids) project that can be controlled from a smartphone. A Raspberry Pi Zero and camera are mounted to the carriage, [along with

> I think it's amazing the sense of unity a community can feel when they get their hands on a junk part

a] built-in motor driver and LiPo battery.

I enjoyed using the motor driver to control acceleration and deceleration, and the design of the web UI.

What differences have you noticed in the Japanese maker community compared to the USA/UK?

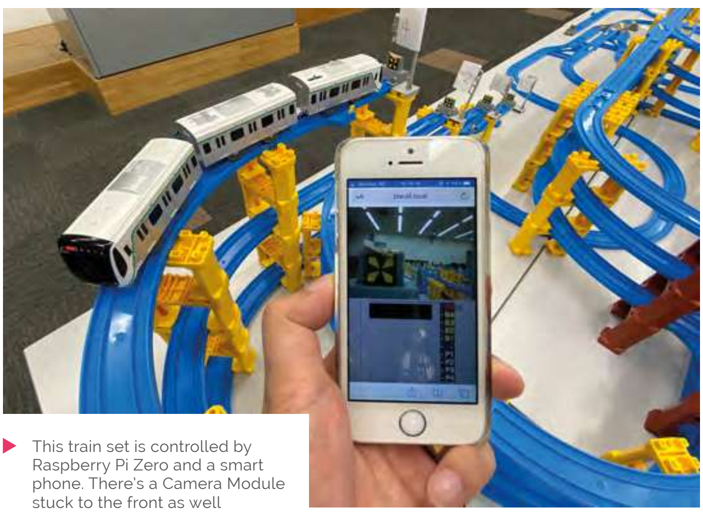
I don't know much about overseas maker communities (or Japanese communities even), but when I saw maker events in US or the UK, I found there were more huge (literally!) projects than in Japan. It might be hard for us to create huge projects in Japan because, in many cases, there is not enough space in our house to work in.

Furthermore, Japan has its own certification system for power supplies and radio waves, and it often takes a long time for foreign products to become available in Japan. Sometimes they cannot be used in Japan because the manufacturer often does not obtain certification in Japan. Compared to the certification system in US or UK, I assume it makes things less flexible for Japanese makers' projects which use radio waves…

Apart from our country's system, I think it's amazing the sense of unity a community can feel when they get their hands on a junk part. In early 2021, it was very popular for Japanese makers on Twitter to take out a 640×48 pixel LCD that was built into a children's toy and analyse it. No matter if they knew each other or not, makers brought information they had to Twitter and GitHub, and made controlling the LCD possible in a few weeks. That was really amazing. It is available for Raspberry Pi as well. I followed those instructions, bought the toy, and made it work too.

Akkie's first Raspberry Pi, as posted to his Twitter account in 2012

Translation by Satoka Fujita



This train set is controlled by Raspberry Pi Zero and a smart phone. There's a Camera Module stuck to the front as well

See Akkie's work

"[Over the last year], there has been no online event for Open Source Conference (OSC: **ospn.jp**). I usually exhibit and introduce Raspberry Pi projects, and give a presentation as Japanese Raspberry Pi Users Group at the OSC's events. I also showed projects at Japan's Maker Faire events.

"I post about projects and [the] making process on my blog (**magpi.cc/akkieblog**) and Mastodon (**magpi.cc/akkie**)."

MagPi Monday

Amazing projects direct from our Twitter!

Every Monday we ask the question: have you made something with a Raspberry Pi over the weekend? Every Monday, our followers send us amazing photos and videos of the things they've made.

Here's a selection of some of the awesome things we got sent this month – and remember to follow along at the hashtag #MagPiMonday!!

01. Pet-cam number one this month, with a cute dog to keep track of

02. A useful range sensor! And a clever way to solve a voltage problem

03. Chickens count as pets, right? A great use of a simple automated Raspberry Pi project

04. Pet-cam the third, with a small cat napping in the corner and tweeting in private

05. Litter collection of the future maybe?

06. This would not look out of place on the robot that Sly is cutting from *Rocky IV*

07. This is a very cool upcycling project. Be careful working with a big computer PSU, though!

08. Robot mower updates are always fun – this one keeps getting better

09. A great start to Halloween decorations from early October

10. This is a very impressive synth idea brought to life!

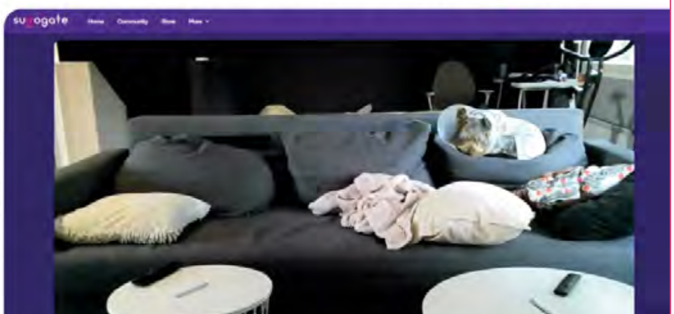
11. Using some clever code, CycOb has managed to create virtual bumpers so that this robot knows when it has hit an obstacle by how much power the motor is using

01

Surrogate
@SurrogateTV

Replying to @TheMagPi

A Raspberry Pi doggie cam to check on the dog and give her snacks at a press of a button. 😋🐶

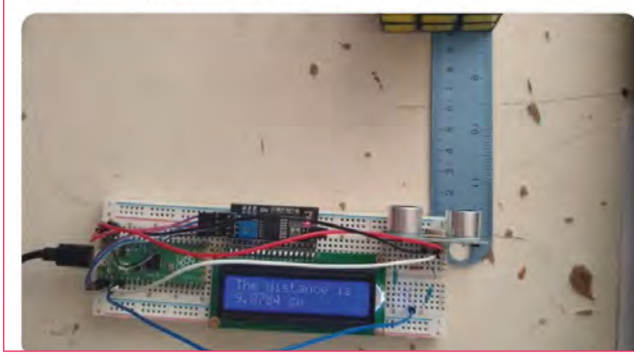


02

Frank
@Frank76139220

Replying to @TheMagPi

Managed to combine the ultrasonic distance sensor with the LCD through my Pico. Had to use a voltage divider as my HC-SR04 is neither + nor P.



03

Cédric de l'Epine
@Spiderland_

Replying to @TheMagPi

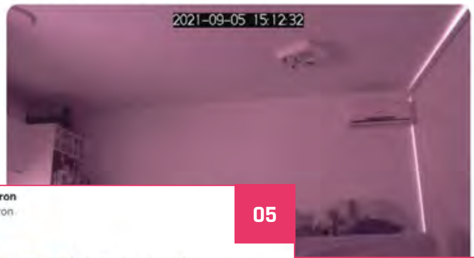
My chicks doorman :)
IR cam and gate control (open / close the coop via cron and web)

04

Naranjú
@yuudj

Replying to @TheMagPi

Python + MQTT + TweePy : Pet cam. Send commads vía MQTT,takes picture or animate gif. Post it in a privarse tweeter account.



05

TheAmplituhedron
@ThAmplituhedron

Replying to @TheMagPi

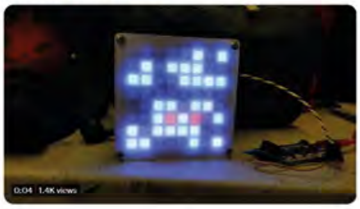
Based on @Raspberry_Pi 4, I designed this autonomous (lidar) litter detection robot employing an @EdgeImpulse object detection model to recognize and classify litter. It also deploys a video stream and a fall detection system.

06

Brian Corteil
@CannonFodder

Replying to @TheMagPi

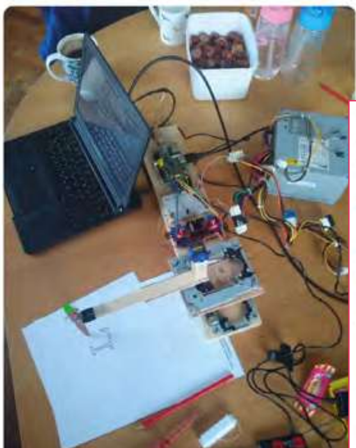
I made a matrix flash LEDs randomly with a @pimoroni Plasma RP2040 LED controller board for my #PiWars Doofus robot in the style of a 50s / 60s movie computer. #MagPiMonday



07

Replying to @TheMagPi

Got my CD-ROM plotter working! Thanks to @homofaciens! Still needs some tweaking...



08

Replying to @TheMagPi

No Halloween edition, but the upcoming Model D the PiMowBot will have a tilt compensated compass based on a 9-axis IMU MPU-9250-module and a more powerful motor driver. I just finished the software tweaks for this over the weekend.

#RaspberryPi #Robotics #Lawnmower

09

Stewart Watkiss
@stewartwatkiss

Replying to @TheMagPi

Raspberry Pi pixel-server to control RGB LEDs / NeoPixels using a web browser.

Also works with smartphone and home automation.

youtu.be/s5Evig_VRdE

It will be used for lighting up my driveway at Halloween, but remember: a Raspberry Pi is for life, not just for Halloween

youtube.com
Raspberry Pi Wireless NeoPixel server for RGB LEDs
Run a wireless NeoPixel / Pixelstrip server on a Raspberry Pi. Using Python, Flask and WS281x library turns a Raspberry ...

10

Replying to @TheMagPi

Nothing for Halloween, but this is looking like it is to have it's scary moments...

diyelectromusic.wordpress.com/2021/10/08/bar...

#MagPiMonday

diyelectromusic.wordpress.com
"Bare Metal" Raspberry Pi Synth

11

GoPiGo3_Stalled_Wheels.mov

Best of the rest!

Amazing projects that we were contacted about this month

PIHUNTER

> magpi.cc/pihunter

We got an email from Gabriel Simches, a cyber security analyst, about his Raspberry Pi threat-hunting device, which he has excellently called piHunter. He reckons it will work at home and in a "small to mid-sized office" as well.

"Set up a Raspberry Pi to collect full-packet captures on your network, then index, normalise, and search through your data to find anomalies," Gabriel writes on GitHub. "All the data and tools needed to deep-dive into any irregularity or weird activity is ready to go! A good threat-hunting device has both GUI tools and multiple Living off the Land (LotL) tools."

Here are the tools it runs:

Arkime (Moloch)	Suricata	Zeek (Bro)	Elasticsearch	Kibana
Filebeat	Winlogbeats (TBD)	RITA	Run-RITA-Run	tcpdump
tshark	prads	strings	yara	python
scapy	PowerShell	foremost	Nmap	

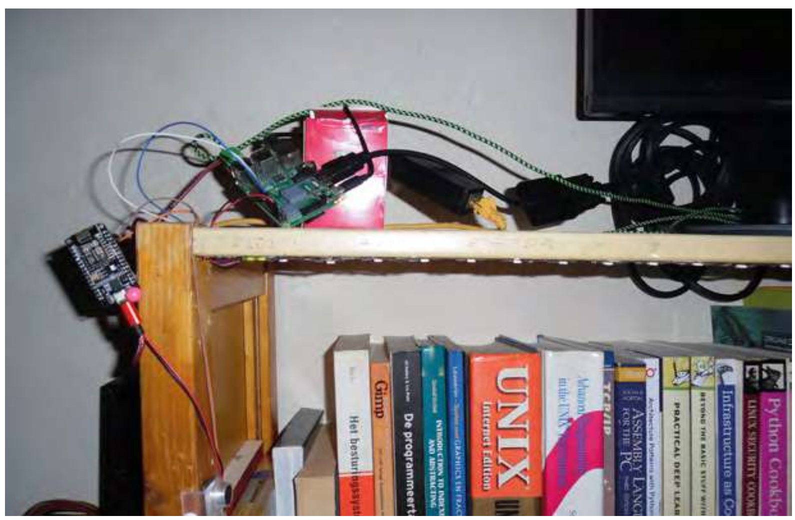
INTERACTIVE BOOKSHELF

> magpi.cc/iotlibrary

"The Internet of Things landscape is riddled with commercial companies that want to gather as much data as they can, while I think a library thrives on privacy," writes Willem Jan in his blog about his interactive bookshelf idea. Using a series of sensors and LEDs, Willem thinks he can make finding and discovering books more fun.

The blog goes into his research on sensors, microcontrollers, and microcomputers, before giving a practical build for your bookshelf.

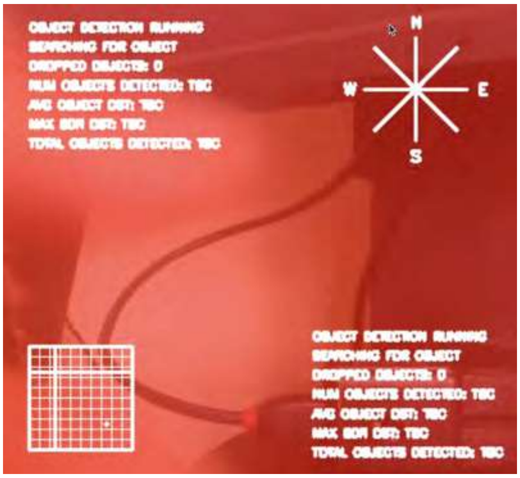
"Although this blog is very technical, I will try to inform a general audience about the possibilities that the Internet of Things has to offer."



TERMINATOR HUD

> magpi.cc/termhud

Ever wanted to see like the Terminator? Maybe not but it's a cool idea for a video filter that looks great as well. It's all written in Python for Raspberry Pi Camera Modules, and works best on Raspberry Pi 4 as it needs a bit of oomph to get it working.



TRACKING PLANES IN BRAZIL

> magpi.cc/coffeeradiopod

We got an email from Martin Butera who hosts a podcast in Brazil's capital, Brasilia, about radio and coffee. While talking to Thiago Pereira Machado, he mentioned how he used Raspberry Pi to track aircraft, a cool and common project for English speakers but not one we've seen in Portuguese before.

Crowdfund **this!**

Raspberry Pi projects you can crowdfund this month

Mini Pupper

A cute quadrapedal robot based on Raspberrry Pi, it's perfect for learning about robotics with it's incredible 12 degrees-of-freedom and open source software including the Robot Operating System, OpenCV for computer vision, and SLAM for mapping its surroundings.

▶ **kck.st/3zVVaZA**

Little Hackers

Learning to code with less screen time was something Google Engineer Brandon Tory was worried about, so he created a book to allow you to do just that with help from his six-year-old son

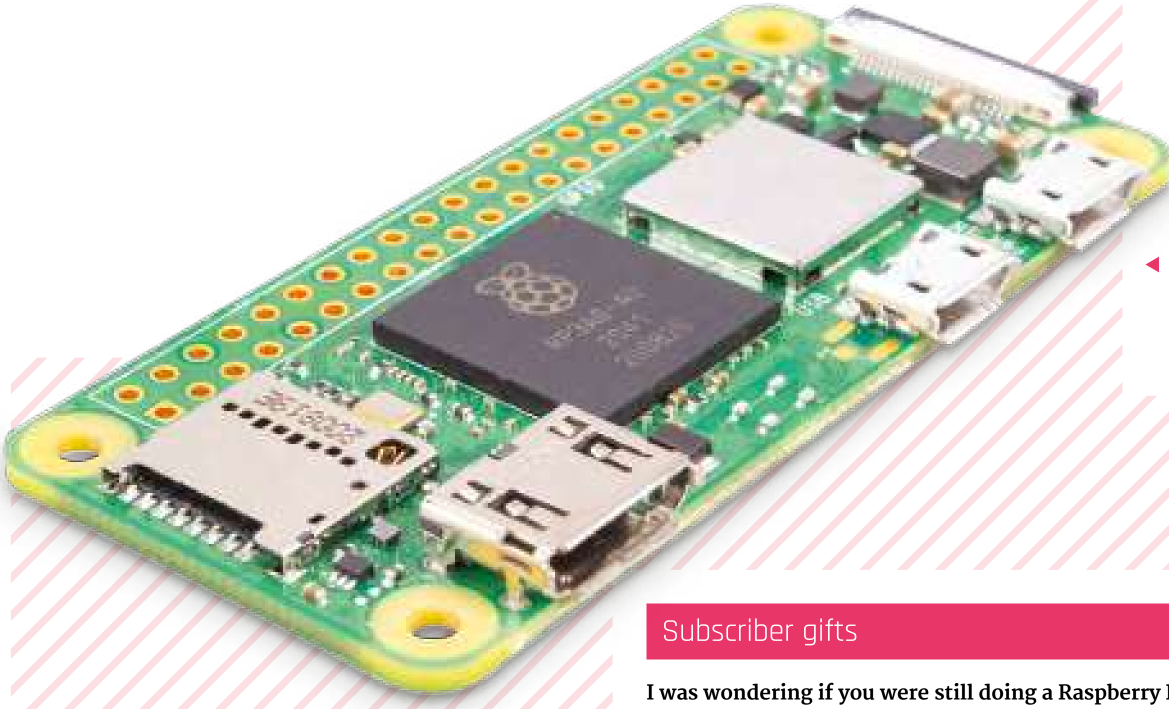
▶ **kck.st/3zOfvQr**

CROWDFUNDING A PROJECT?

If you've launched a Raspberry Pi-related project, let us know!

magpi@raspberrypi.com

Your Letters



Raspberry Pi Zero 2 is here, and we gave one to all current subscribers. Get yours with a new subscription

Subscriber gifts

I was wondering if you were still doing a Raspberry Pi Zero with your subscriptions? Do subscribers get anything else as well with their subscription? I've been meaning to get one for ages.

Christof via Twitter

As you may have noticed by now, the subscription offer has been upgraded to include a Raspberry Pi Zero 2. You can find more info about that on page 32. As for gifts, current subscribers also received a Zero 2 in the post. Subscribing is the best way to make sure you get any of the special items we give away!

▲ A lot of people love to share what they're working on via Twitter

▲ Make sure to check out our spooky project upgrades feature in issue 110 for some last-minute ideas

Project submissions

If I have a Raspberry Pi project I'd like you to see, where is the best place to send it?

I think it would be a great fit for *The MagPi*.

Daisy via email

We're more than happy to look at any Raspberry Pi, Raspberry Pi Pico, or RP2040 projects that you may have done, and emailing is the best way to show us. Otherwise, contacting us on Facebook, or showing us on #MagPiMonday on Twitter is a good start! Sometimes, if we see it online, we'll also contact you about it. Whether it becomes a project showcase, a mention in This Month in Raspberry Pi, or a retweet, we love to see and share what you're making!

Nightmare before Christmas

Will you be showing off Halloween projects again this year from the community? I've got some really fun plans and I was wondering whether I should send them to you! I hope you like being scared…

Phil via Facebook

We absolutely will if you share it with us! Due to the peculiarities of creating a magazine, this November issue will come out a few days before Halloween, and the issue after Halloween has happened will be the December issue – which will have Christmas stuff in it. Don't worry, though: we'll print them any time of the year.

Speaking of which, don't forget to send us your Christmas projects too! Although they'll probably end up in the February issue.

Contact us!

- Twitter **@TheMagPi**
- Facebook **magpi.cc/facebook**
- Email **magpi@raspberrypi.com**
- Online **forums.raspberrypi.com**

WIN ONE OF FIVE WAVESHARE E-INK DISPLAY PHATS

WITH 20 HEATSINKS FOR RUNNERS-UP!

IN ASSOCIATION WITH THE PI HUT

This 2.13" E-Ink display is perfect for the Raspberry Pi Zero 2 W that you can get free with your twelve-month subscription to *The MagPi*. We have five to give away, and 20 runner-up prizes of a heatsink for Zero 2 as well.

Head here to enter: magpi.cc/win | Learn more: magpi.cc/einkdisplayphat

Terms & Conditions

Competition opens on **27 October 2021** and closes on **26 November 2021**. Prize is offered to participants worldwide aged 13 or over, except employees of the Raspberry Pi Foundation, the prize supplier, their families, or friends. Winners will be notified by email no more than 30 days after the competition closes. By entering the competition, the winner consents to any publicity generated from the competition, in print and online. Participants agree to receive occasional newsletters from The MagPi magazine. We don't like spam: participants' details will remain strictly confidential and won't be shared with third parties. Prizes are non-negotiable and no cash alternative will be offered. Winners will be contacted by email to arrange delivery. Any winners who have not responded 60 days after the initial email is sent will have their prize revoked. This promotion is in no way sponsored, endorsed or administered by, or associated with, Instagram or Facebook.

CHRISTMAS

Incredible makes for the festive season!

THE MAGPI **#112**
ON SALE **25 NOVEMBER**

Plus!

Raspberry Pi Zero 2 W projects

Upgrade a Super 8 Camera

Build an intruder alarm

DON'T MISS OUT!

magpi.cc/subscribe

TWITTER	@TheMagPi
FACEBOOK	fb.com/MagPiMagazine
EMAIL	magpi@raspberrypi.com

EDITORIAL

Editor
Lucy Hattersley
lucy@raspberrypi.com

Features Editor
Rob Zwetsloot
rob@raspberrypi.com

Sub Editor
Nicola King

ADVERTISING

Charlotte Milligan
charlotte.milligan@raspberrypi.com
+44 (0)7725 368887

DESIGN

criticalmedia.co.uk

Head of Design
Lee Allen

Designers
Lucy Cowan, Sam Ribbits

Illustrator
Sam Alder

CONTRIBUTORS

David Crookes, PJ Evans, Satoka Fujita, Rosemary Hattersley, Edwin Jones, Nicola King, Phil King, KG Orphanides

PUBLISHING

Publishing Director
Russell Barnes
russell@raspberrypi.com

Director of Communications
Liz Upton

CEO
Eben Upton

DISTRIBUTION

Seymour Distribution Ltd
2 East Poultry Ave,
London EC1A 9PT
+44 (0)207 429 4000

SUBSCRIPTIONS

Unit 6 The Enterprise Centre
Kelvin Lane, Manor Royal,
Crawley, West Sussex, RH10 9PE
+44 (0)1293 312193
magpi.cc/subscribe
magpi@subscriptionhelpline.co.uk

Tapping into tech

Intriguing early encounters with computing led to **Rosie Hattersley's** career writing about them

I first encountered the concept of computers when my visiting aunt reached into her bag and triumphantly produced lengths of stiff cardboard with seemingly random bits punched out. Whereas my mum had been one of the earliest female students of cybernetics, her sister had gone into teaching maths. Both were perfectly placed to encounter programming 1970s-style, and my mum spent two happy years as a Fortran programmer.

School meant Casio watches and LED displays; sometimes built into pleasingly flexible, twangable rulers; sometimes embedded into the shaft of a pen. Resetting the time and date was the ideal excuse to linger in a classroom while chatting to (aka flirting with) nerdy, but nice, schoolmates. Almost without fail, an overlooked friend would interject with a rude word spelled out on their fancy scientific calculator screen.

The arrival of two BBC Micros in the school library caused a stir. As a wannabe journalist, I became the editor of the newspaper we put together as a class. Getting to use one at school meant taking business and information studies so I could get at the Acorn computers in the commerce suite. Unfortunately, I had to spend the first half-term learning to touch-type – anathema for an impatient and less than dextrous teenager.

Commodore coding

Thankfully, a brilliant Commodore 64 made it into our home. I spent hours learning to play a musical keyboard which sat above the computer's own keyboard, and typing in game code from books. I taught myself BASIC and set up modest databases to organise my growing record collection, but was often ousted from the C64 by my games-hungry brother, keen to thrash another joystick in the service of Daley Thompson's Decathlon.

> "The arrival of two BBC Micros in the school library caused a stir"

Our Dad spent an apocryphal £3000 on a Elonex 386 laptop, heavier even than the Smith Corona typewriter with the three-page text memory that I would write my university essays on.

My first computer was a second-hand Mac, so I could brush up the QuarkXpress and Photoshop skills I used in my first job in publishing, indirectly taking me to a writing role at home computing title *PC Advisor* where I got to try out, review, and write about using hundreds of gadgets, games, apps and programs and, eventually, learn about an intriguing product known as Raspberry Pi and its aim to put computers and computing skills in the reach of the world. The global success of maker faires, Raspberry Jams, Pi Wars, Code Clubs and dojos, as well as the dozens of individual projects that I've been lucky enough to learn about first hand and write about for *The MagPi*, show that Raspberry Pi has brilliantly delivered.

AUTHOR
Rosie Hattersley

Rosie Hattersley has been writing about the joy of tech for the past two decades and (occasionally) still misses QuarkXpress.

@RosieHattersley